AXI Block RAM (BRAM) Controller v4.0

LogiCORE IP Product Guide

Vivado Design Suite

PG078 September 30, 2015

XILINX
ALL PROGRAMMABLE™

XILINX
ALL PROGRAMMABLE™

Table of Contents

Appendix B: Verification, Compliance, and Interoperability

Appendix C: Debugging

Appendix D: Additional Resources and Legal Notices

Introduction

The LogiCORE™ IP AXI Block RAM (BRAM) Controller is a soft IP core for use with the Xilinx Vivado™ Design Suite. The core is designed as an AXI Endpoint slave IP for integration with the AXI interconnect and system master devices to communicate to local block RAM. The core supports both single and burst transactions to the block RAM and is optimized for performance.

Features

- AXI4 (memory mapped) slave interface
- Low latency memory controller
- Separate read and write channel interfaces to utilize dual port FPGA BRAM technology
- Option to arbitrate read and write data for use with a single port of block RAM (in AXI4 and AXI4LITE modes)
- Configurable BRAM data width (32-, 64-, 128-, 256-, 512-, and 1024-bit) (equals AXI slave port data width size)
- Supports memory sizes up to a maximum of 2 MBytes (byte size 8 or 9)
- Supports INCR burst sizes up to 256 data transfers, and WRAP bursts of 2, 4, 8, and 16 data beats
- Supports AXI narrow and unaligned write burst transfers
- Compatible with Xilinx AXI Interconnect
- Supports two-deep address pipelining on each read and write channel (order must be maintained)
- Reduced footprint option for AXI4-Lite
- Supports of both Internal and External modes of BRAM block instances
- Optional ECC support with 32-, 64- or 128-bit BRAM data widths
    - AXI4-Lite register interface for status and control of ECC operations
    - Available interrupt output signal for ECC error detection
- Hamming code and HSIAO algorithm support for 32, 64-bit BRAM data widths and HSIAO algorithm support for 128-bit BRAM data width

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family[1]	UltraScale™ Architecture, Zynq ®-7000, 7 Series
Supported User Interfaces	AXI4, AXI4-Lite
Resources	See Performance and Resource Utilization web page.
Provided with Core	
Design Files	VHDL
Example Design	VHDL
Test Bench	VHDL
Constraints File	XDC
Simulation Model	Not Provided
Supported S/W Driver[2]	Standalone
Tested Design Flows[3]	
Design Entry	Vivado Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx at the Xilinx Support web page	

Notes:

1. For a complete listing of supported devices, see the Vivado IP Catalog.
2. Standalone driver details can be found in the SDK directory (*<install_directory>*/doc/usenglish/xilinx_drivers.htm). Linux OS and driver support information is available from the Xilinx Wiki page.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The AXI BRAM Controller core can be configured such that a single port to the BRAM block or both ports to the BRAM block are utilized in either an AXI4 or AXI4-Lite controller configuration. The AXI BRAM Controller IP can be configured with ECC functionality on the datapath with an available external ECC register set via a second AXI4-Lite control port connection. Figure 1-1 and Figure 1-2 illustrate the top-level port connections and main modules of the AXI BRAM Controller IP core. Several instantiation modes exist based on interface connection and BRAM module port utilization.

Figure 1-1 illustrates the connections when the AXI BRAM core is configured for an AXI4-Lite mode to the BRAM block. A single port utilization to the BRAM block or a dual port mode to the BRAM block can be utilized (via a parameter setting). Figure 1-2 illustrates the generated HDL core for supporting a AXI4 transaction interface. Single port usage to the BRAM block can be configured over an enhanced performance setting in a dual port configuration.

All communication with AXI master devices is performed through a five channel AXI interface. All write operations are initiated on the Write Address Channel (AW) of the AXI bus which specifies the type of write transaction and the corresponding address information. The Write Data Channel (W) communicates all write data for the single or burst write operations. The Write Response Channel (B) is used as the handshaking or response on the write operation.

On read operations, the Read Address Channel (AR) communicates all address and control information when the AXI master requests a read transfer. The AXI slave AXI BRAM Controller IP responds on the Read Address Channel (AR) when the read operation can be processed. When the read data is available to send back to the AXI master, the Read Data Channel (R) translates the data and status of the operation.

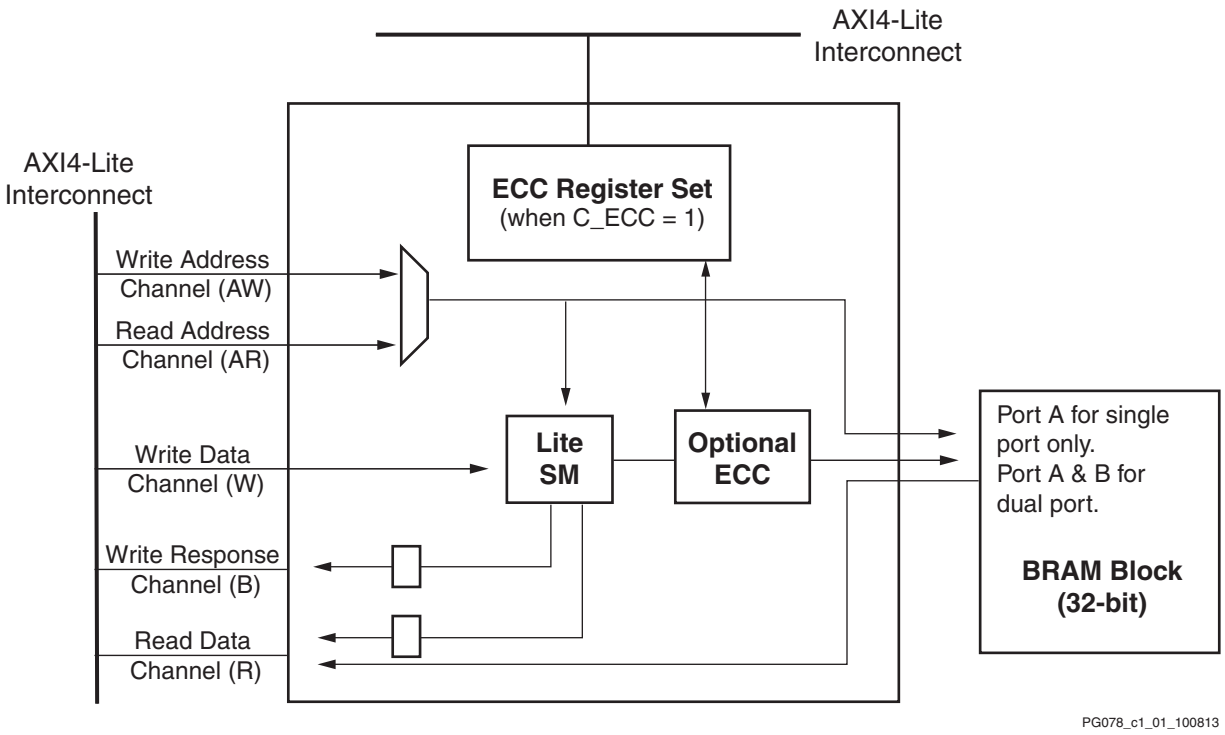

Figure 1-1: **AXI4-Lite BRAM Controller Block Diagram**

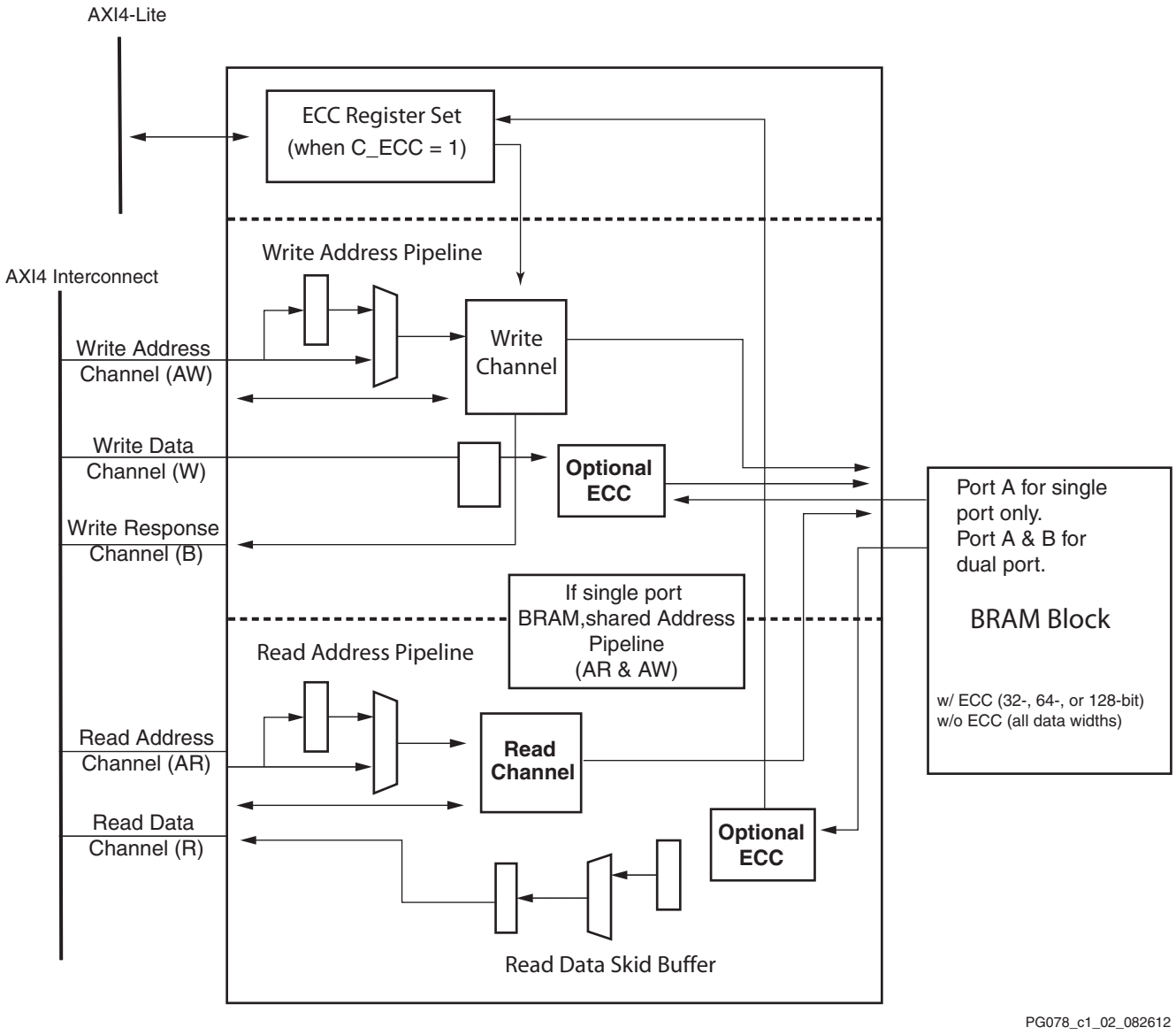

Figure 1-2: **AXI4 BRAM Controller Block Diagram**

ECC Description

You can enable ECC functionality in the IP core. ECC allows an AXI master to detect and correct single and detect double bit errors in the BRAM block. The ECC status and control are accessible via an additional AXI4-Lite control interface on the AXI BRAM Controller core. The ECC functionality can be enabled regardless of dual or single port BRAM access. ECC is enabled by configuring the design parameter, C_ECC = 1. The ECC is available only when the BRAM block is configured to a 32-, 64-, or 128-bit data width. Hamming code and HSIAO algorithms are available for 32 and 64-bit BRAM data widths. For 128-bit BRAM data width configurations, only HSIAO algorithm is available.

Feature Summary

AXI4 Compatibility

The AXI BRAM Controller IP core is compliant to the AMBA® AXI4 interface specifications listed in References in Appendix D. The AXI BRAM Controller core includes these features and exceptions:

- Support for 32-, 64-, 128-, 256-, 512-, and 1024-bit BRAM data widths
- Support for all AXI4 burst types and sizes
  - AXI BRAM Controller handles FIXED bursts as INCR type burst operations (no FIFO burst type capability in the BRAM core)
  - 16 beats for WRAP bursts
  - 16 beats for FIXED bursts (treated as INCR burst type)
  - 256 beats for INCR burst (without exclusive access)
- Support for burst sizes that are less than the width of the block RAM, for example, narrow bursts. Data transfers are on different byte lanes for each beat of the burst.
- AXI user signals are not necessary or supported
- The AXI BRAM Controller executes all transactions in order regardless of thread ID value. No read reordering or write reordering is implemented.
- No caching or buffering functionality is supported in the AXI BRAM Controller

AXI4-Lite Support

As per the AXI specification, the AXI BRAM Controller supports all requests from an AXI4-Lite master or AXI4-Lite Interconnect. The core can be configured for optimized FPGA resource usage and BRAM port utilization in this mode. The AXI4-Lite mode only supports a 32-bit AXI data bus width and single data beat transfers. No unaligned, narrow, or burst types transfers are acknowledged by the AXI BRAM Controller IP core when configured in this mode.

The AXI4-Lite IP core can be configured as dual port to the block RAM or configured to arbitrate to a single port of block RAM at the AR and AW AXI interfaces. In this mode (C_SINGLE_PORT_BRAM = 1), only one active operation is allowed at a time in the AXI BRAM Controller core.

BRAM Interface

The BRAM interface is optimized to provide the highest performance interface to the FPGA BRAM module. The dual-port capability of the Xilinx BRAM technology is utilized in this

design (when configured in dual-port mode). Port A of the BRAM module is designated as the write port, while Port B of the BRAM module is designated as the read port.

For decreased size, the AXI BRAM Controller can be configured in a single port utilization to the BRAM module. In this case, Port A is used for both read and write operations to block RAM. To reduce resource utilization and limit any potential impacts to latency, any collisions between the read and write ports of the block RAM are not detected by the AXI BRAM Controller.

Systems using 7 series FPGAs that are generated in IP Integrator have a bank of block RAM based on incorporating the RAMB18E1 or the RAMB36E1. The block RAM is configured using the Block Memory Generator IP core. The BRAM block instantiation is not included in the AXI BRAM Controller IP in an IP Integrator system topology.

Instantiating the AXI BRAM Controller from the Vivado IP Catalog as a stand-alone IP core requires the BRAM components be instantiated separately with the Block Memory Generator tool.

Supported Memory Size

The AXI BRAM Controller supports memory sizes up to a maximum of 2 MBytes (byte size 8 or 9). Table 1-1 shows the supported memory width and depth of AXI BRAM Controller.

Table 1-1: **Support Memory Sizes**

Data Width (Bits)	Memory Depths (Words)
32	1k,2k,4k,8k,16k,32k,64k,128k
64	1k,2k,4k,8k,16k,32k,64k,128k
128	1k,2k,4k,8k,16k,32k,64k,128k
256	1k,2k,4k,8k,16k,32k,64k
512	1k,2k,4k,8k,16k,32k
1024	1k,2k,4k,8k,16k

ECC Support

ECC support can be enabled in the AXI BRAM Controller IP core by setting the design parameter, C_ECC = 1. ECC is only supported for BRAM data widths of 32-, 64-, or 128-bit configurations. More information is described in ECC, page 61.

Applications

The AXI BRAM Controller core is designed to integrate in an AXI system via the AXI Interconnect topology to provide multiple masters access to block RAM. The AXI BRAM Controller is an endpoint slave IP core to be attached as a slave AXI device on the AXI

Interconnect. The AXI Interconnect allows the AXI BRAM Controller to be optimized for latency and reduce the resources utilized. The AXI Interconnect performs any mismatching of master data widths to the size of the block RAM and the AXI BRAM Controller core.

TIP: *Xilinx recommends that the AXI BRAM Controller's AXI4 interface data width match the native width of the AXI Interconnect to which it is attached.*

The AXI Interconnect performs the decode of the master address prior to presenting any operation to the AXI BRAM Controller core. Additional information on the Xilinx AXI Interconnect is available in References in Appendix D.

The connection of the AXI BRAM Controller core into an example system topology is shown in Figure 1-3 for both single and dual port BRAM module configurations.

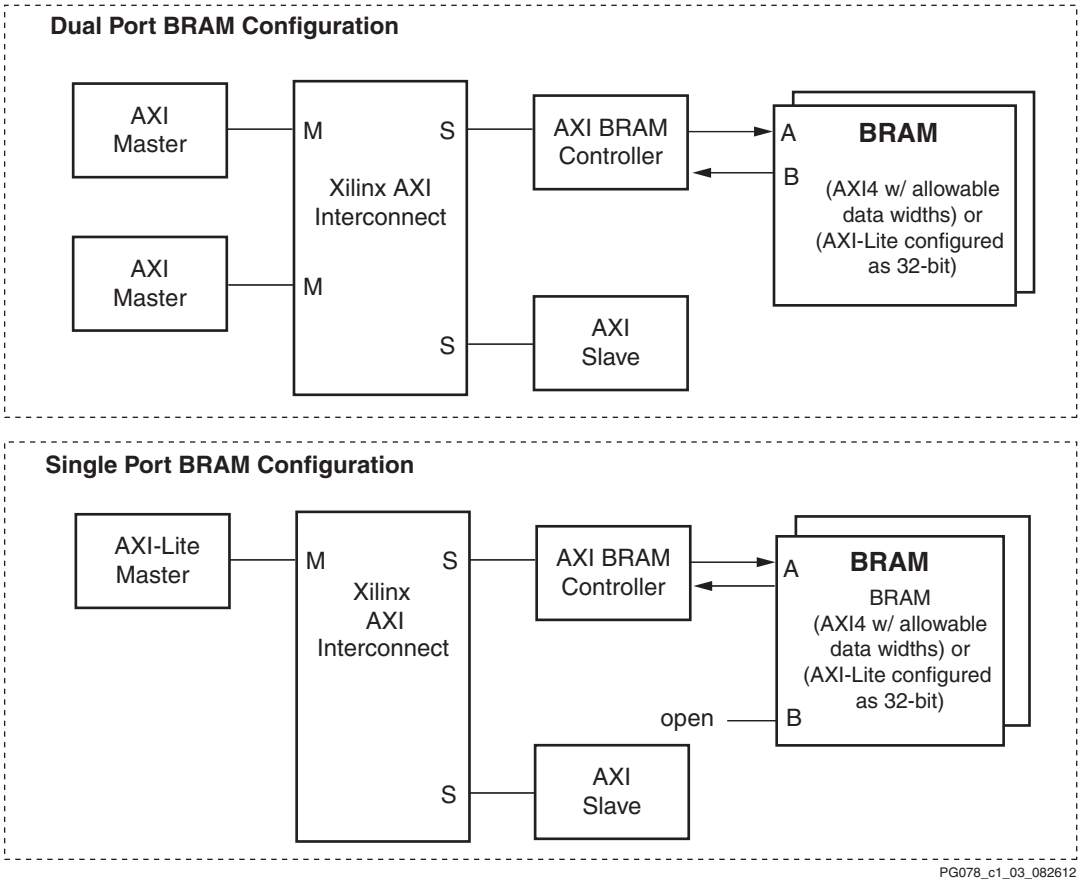

Figure 1-3: **AXI System Configuration**

Licensing and Ordering Information

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx Vivado Design Suite under the terms of the Xilinx End User License.

Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

Standards

The AXI BRAM Controller adheres to the AMBA® AXI4 Interface standard (see *ARM® AMBA AXI Protocol Specification, Version 2.0 ARM IHI 002C*) [Ref 1].

Performance

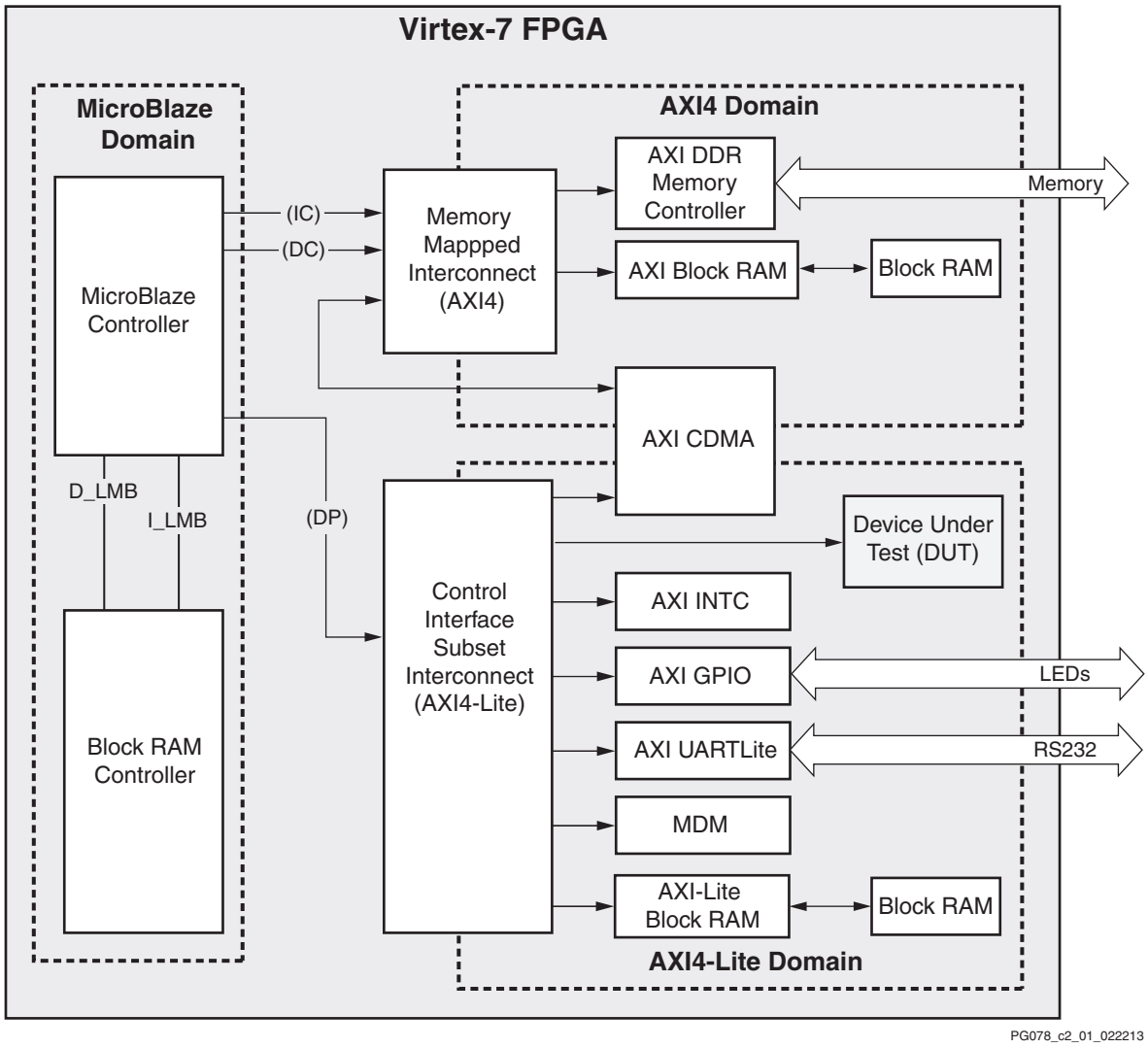

Figure 2-1: **Virtex-7 FPGA System with the AXI BRAM Controller as the DUT**

Maximum Frequencies

The target FPGA was filled with logic to drive the LUT and BRAM utilization to approximately 70% and the I/O utilization to approximately 80%. Using the default tool options and the slowest speed grade for the target FPGA, the resulting target F_{MAX} numbers are shown in Table 2-1.

IMPORTANT: *The maximum frequency values for UltraScale™ architecture and Zynq™-7000 devices are expected to be similar to Artix-7 and Kintex-7 device numbers.*

Table 2-1: **System Performance**

Family	Speed Grade	F_{max} (MHz)	
		AXI4	AXI4-Lite
Virtex-7	-1	200	180
Kintex-7		200	180
Artix-7		150	120
Virtex-7	-2	240	200
Kintex-7		240	200
Artix-7		180	140
Virtex-7	-3	280	220
Kintex-7		280	220
Artix-7		200	160

The target F_{MAX} is influenced by the exact system and is provided for guidance. It is not a guaranteed value across all systems.

Latency

There is single-clock latency between the channel valid and response signals.

Throughput

The nominal throughput is one read or write access every clock cycle. The only exceptions are as follows:

- When performing a byte or half word write while the ECC is enabled.
- When an access is ongoing on another port while using multiple ports.

Resource Utilization

For details about resource utilization, visit Performance and Resource Utilization web page.

Port Descriptions

The AXI BRAM Controller I/O signals are listed and described in Table 2-2.

Table 2-2: **I/O Signal Description**

Signal Name	Interface	Signal Type	Init Status	Description
Global Signals				
s_axi_aclk	S_AXI, S_AXI_CTRL	I		AXI Bus Clock.
s_axi_aresetn	S_AXI, S_AXI_CTRL	I		AXI active-Low reset.
ecc_interrupt		O	'0'	Interrupt to signal ECC error condition: Signal unused when C_ECC = 0.
ecc_ue		O	'0'	ECC uncorrectable error output flag: The behavior of this flag is described in ECC, page 61. Signal unused when C_ECC = 0.
AXI Write Address Channel Signals (AW)				
s_axi_awid [c_s_axi_id_width-1:0]	S_AXI (AW)	I		AXI address write ID. Only generated when the ID width is set greater than 0.
s_axi_awaddr [c_s_axi_addr_width-1:0]	S_AXI (AW)	I		AXI write address
s_axi_awlen [7:0]	S_AXI (AW)	I		AXI address write burst length: The burst length gives the exact number of transfers in a burst. Determines the number of data transfers associated with the write address.
s_axi_awsize [2:0]	S_AXI (AW)	I		AXI address write burst size: Indicates the size of each transfer in the burst. Byte lane strobes determine exactly which byte lanes to update.
s_axi_awburst [1:0]	S_AXI (AW)	I		AXI address write burst type: Indicates the burst type, coupled with the size information, details how the address for each transfer within the burst is calculated.
s_axi_awlock [2:0]	S_AXI (AW)	I		AXI write address lock signal: Provides information about atomic operation transfers and barrier transactions. Unused at this time, but listed here for future implementation and support.
s_axi_awcache [4:0]	S_AXI (AW)	I		AXI write address cache control signal: Provides information about bufferable, cacheable, and allocation attributes. Unused at this time, but listed here for future implementation and support.
s_axi_awprot [3:0]	S_AXI (AW)	I		AXI write address protection signal. Unused at this time, but listed here for future implementation and support.

Table 2-2: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
s_axi_awvalid	S_AXI (AW)	I		AXI write address valid. Indicates that the valid write address and control information is available. '0' = address and control NOT available '1' = address and control data is available The address and control information remains stable until the acknowledge signal (S_AXI_AWREADY) is asserted (active-High).
s_axi_awready	S_AXI (AW)	O	'0'	AXI write address ready. Indicates that the AXI BRAM Controller has accepted the write channel address and associated control signals. '0' = AXI BRAM Controller not ready '1' = AXI BRAM Controller is ready
AXI Write Data Channel Signals (W)				
s_axi_wdata [c_s_axi_data_width-1:0]	S_AXI (W)	I		AXI write data.
s_axi_wstrb [c_s_axi_data_width/8-1:0]	S_AXI (W)	I		AXI write data strobes. Indicates which bytes lanes to update in block RAM. Each byte strobe correlates to a byte other write data bus. WSTRB[n] corresponds to WDATA[(8*n) + 7:(8*n)].
s_axi_wlast	S_AXI (W)	I		AXI write data last signal. Indicates the last transfer in a write burst.
s_axi_wvalid	S_AXI (W)	I		AXI write data valid. Indicates that valid write data and strobes are available on the write data channel. '0' = write data and strobes unavailable '1' = write data and strobes available and valid
s_axi_wready	S_AXI (W)	O	'0'	AXI write data ready. indicates that the AXI BRAM Controller is ready to accept the write data and strobes. '0' = AXI BRAM Controller not ready '1' = AXI BRAM Controller is ready
AXI Write Response Channel Signals (B)				
s_axi_bid [c_s_axi_id_width-1:0]	S_AXI (B)	O	Zeros	AXI write data response ID. Identification tab of the write response. BID matches the AWID value of the write transaction to which the AXI BRAM Controller is responding. Only generated when the ID width is set greater than 0.

Table 2-2: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
s_axi_bresp [1:0]	S_AXI (B)	O	Zeros	AXI write response. Indicates the status of the write transaction. The supported responses of the AXI BRAM Controller are OKAY (SLVERR and EXOKAY to be supported in a future release).
s_axi_bvalid	S_AXI (B)	O	'0'	AXI write response valid. Asserted by the AXI BRAM Controller to indicate a valid write response is available. '0' = write response not available '1' = write response is available
s_axi_bready	S_AXI (B)	I		Write response ready. Indicates the master requesting the write operation can accept the response information. '0' = AXI master not ready '1' = AXI master is ready
AXI Read Address Channel Signals (AR)				
s_axi_arid [c_s_axi_id_width-1:0]	S_AXI (AR)	I		AXI address read ID. Identification tag for the read address group of signals. Only generated when the ID width is set greater than 0.
s_axi_araddr [c_s_axi_addr_width-1:0]	S_AXI (AR)	I		AXI read address. The address provides the initial address of the read burst transaction. Only the start address of the burst is provided and the control signals indicate the address detail and calculation for each transfer in the burst.
s_axi_arlen [7:0]	S_AXI (AR)	I		AXI address read burst length. The burst length gives the exact number of transfers in a burst. Determines the number of data transfers associated with the read address.
s_axi_arsize [2:0]	S_AXI (AR)	I		AXI address read burst size. Indicates the size of each transfer in the burst.
s_axi_arburst [1:0]	S_AXI (AR)	I		AXI address read burst type. Indicates the burst type, coupled with the size information, details how the address for each transfer within the burst is calculated.
s_axi_arlock [2:0]	S_AXI (AR)	I		AXI read address lock signal. Provides information about atomic operation transfers. Unused at this time, but listed here for future implementation and support.
s_axi_arcache [4:0]	S_AXI (AR)	I		AXI read address cache control signal. Provides information about bufferable, cacheable, and allocation attributes. Unused at this time, but listed here for future implementation and support.

Table 2-2: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
s_axi_arprot [3:0]	S_AXI (AR)	I		AXI read address protection signal. Unused at this time, but listed here for future implementation and support.
s_axi_arvalid	S_AXI (AR)	I		AXI read address valid. Indicates that the valid read address and control information is available. '0' = address and control NOT available '1' = address and control data is available The address and control information remains stable until the acknowledge signal (S_AXI_ARREADY) is asserted (active high).
s_axi_arready	S_AXI (AR)	O	'0'	AXI read address ready. Indicates that the AXI BRAM Controller has accepted the read channel address and associated control signals. '0' = AXI BRAM Controller not ready '1' = AXI BRAM Controller is ready
AXI Read Data Channel Signals (R)				
s_axi_rid [c_s_axi_id_width-1:0]	S_AXI (R)	O	Zeros	AXI read data response ID. Identification tab of the read response. RID matches the ARID value of the read address transaction to which the AXI BRAM Controller is responding. Only generated when the ID width is set greater than 0.
s_axi_rdata [c_s_axi_data_width-1:0]	S_AXI (R)	O	Zeros	AXI read data.
s_axi_rresp [1:0]	S_AXI (R)	O	Zeros	AXI read response. Indicates the status of the read transfer. AXI BRAM Controller supports the OKAY and SLVERR responses.
s_axi_rlast	S_AXI (R)	O	'0'	AXI read data last signal. Indicates the last transfer in a read burst.
s_axi_rvalid	S_AXI (R)	O	'0'	AXI read valid. Asserted by the AXI BRAM Controller to indicate the required read data is available and the read transfer can complete. '0' = read data not available '1' = read data is available
s_axi_rready	S_AXI (R)	I		Read ready. Indicates the requesting master can accept the read data and response information. '0' = AXI master not ready '1' = AXI master is ready
AXI4-Lite Control Signals (Only Available when C_ecc = 1)				

Table 2-2: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
s_axi_ctrl_awaddr [c_s_axi_ctrl_addr_ width-1:0]	S_AXI_CTRL (AW)	I		AXI4-Lite control write address
s_axi_ctrl_awvalid	S_AXI_CTRL (AW)	I		AXI4-Lite control write address valid. Indicates that the valid write address and control information is available. '0' = address and control NOT available '1' = address and control data is available The address and control information remains stable until the acknowledge signal (S_AXI_LITE_AWREADY) is asserted (active-High).
s_axi_ctrl_awready	S_AXI_CTRL (AW)	O	'0'	AXI4-Lite control write address ready. Indicates that the AXI BRAM Controller has accepted the write channel address and associated control signals. '0' = AXI BRAM Controller not ready '1' = AXI BRAM Controller is ready
s_axi_ctrl_wdata [c_s_axi_ctrl_data_ width-1:0]	S_AXI_CTRL (W)	I		AXI4-Lite control write data.
s_axi_ctrl_wvalid	S_AXI_CTRL (W)	I		AXI4-Lite write data valid. Indicates that valid write data and strobes are available on the write data channel. '0' = write data and strobes unavailable '1' = write data and strobes available and valid
s_axi_ctrl_wready	S_AXI_CTRL (W)	O	'0'	AXI4-Lite control write data ready. indicates that the AXI BRAM Controller is ready to accept the write data and strobes. '0' = AXI BRAM Controller not ready '1' = AXI BRAM Controller is ready
s_axi_ctrl_bresp [1:0]	S_AXI_CTRL (B)	O	Zeros	AXI4-Lite control response. Indicates the status of the write transaction. The supported responses of the AXI BRAM Controller are OKAY & SLVERR.
s_axi_ctrl_bvalid	S_AXI_CTRL (B)	O	'0'	AXI4-Lite control write response valid. Asserted by the AXI BRAM Controller to indicate a valid write response is available. '0' = write response not available '1' = write response is available

Table 2-2: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
s_axi_ctrl_bready	S_AXI_CTRL (B)	I		AXI4-Lite control write response ready. Indicates the master requesting the write operation can accept the response information. '0' = AXI master not ready '1' = AXI master is ready
s_axi_ctrl_araddr [c_s_axi_ctrl_addr_ width-1:0]	S_AXI_CTRL (AR)	I		AXI4-Lite control read address information.
s_axi_ctrl_arvalid	S_AXI_CTRL (AR)	I		AXI4-Lite control read address valid. Indicates that the valid read address and control information is available. '0' = address and control NOT available '1' = address and control data is available The address and control information remains stable until the acknowledge signal (S_AXI_LITE_ARREADY) is asserted (active-High).
s_axi_ctrl_arready	S_AXI_CTRL (AR)	O	'0'	AXI4-Lite control read address ready. Indicates that the AXI BRAM Controller has accepted the read channel address and associated control signals. '0' = AXI BRAM Controller not ready '1' = AXI BRAM Controller is ready
s_axi_ctrl_rdata [c_s_axi_ctrl_data_ width-1:0]	S_AXI_CTRL (R)	O	Zeros	AXI4-Lite control read data.
s_axi_ctrl_rresp [1:0]	S_AXI_CTRL (R)	O	Zeros	AXI4-Lite control read response. Indicates the status of the read transfer. AXI BRAM Controller supports the OKAY and SLVERR responses.
s_axi_ctrl_rvalid	S_AXI_CTRL (R)	O	'0'	AXI4-Lite control read valid. Asserted by the AXI BRAM Controller to indicate the required read data is available and the read transfer can complete. '0' = read data not available '1' = read data is available
s_axi_ctrl_rready	S_AXI_CTRL (R)	I		AXI4-Lite control read ready. Indicates the requesting master can accept the read data and response information. '0' = AXI master not ready '1' = AXI master is ready
BRAM Signals				
bram_rst_a	BRAM	O	'0'	Port A BRAM active-High reset.

Table 2-2: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
bram_clk_a	BRAM	O	'0'	Port A BRAM clock. Connected to AClk with same frequency, same phase alignment.
bram_en_a	BRAM	O	'0'	BRAM Port A (write port) enable signal. Active High.
bram_we_a [(c_s_axi_data_width/8) + c_ecc - 1:0]	BRAM	O	Zeros	BRAM Port A (write port) write enable signal. Active High. Byte wise signal to width of block RAM. If ECC is enabled, the BRAM_WE_A width increases to hold the ECC bits as shown in Table 3-4, page 62.
bram_addr_a [c_bram_addr_width-1:0]	BRAM	O	Zeros	BRAM Port A (write port) address bus. Bus is sized according to data width and based on C_S_AXI_BASEADDR and C_S_AXI_HIGHADDR settings.
bram_wrdata_a [(8 * c_ecc + c_s_axi_data_width)-1:0]	BRAM	O	Zeros	BRAM Port A (write port) write data bus. Size of BRAM write data width is equal to size of AXI slave port connection to the AXI BRAM Controller. If ECC is enabled, the BRAM_WrData_A width increases to hold the ECC bits as shown in Table 3-4, page 62.
bram_rddata_a [(8 * c_ecc + c_s_axi_data_width)-1:0]	BRAM	I		BRAM Port A (read port) read data bus. Size of BRAM read data width is equal to size of AXI slave port connection to the AXI BRAM Controller. If ECC is enabled, the BRAM_RdData_A width increases to hold the ECC bits as shown in Table 3-4, page 62. BRAM_RdData_A port used during the read-modify-write (if needed) when C_ECC = 1. BRAM_RdData_A port also used for read operations when C_SINGLE_PORT_BRAM = 1.
bram_rst_b	BRAM	O	'0'	Port B BRAM active-High reset.
bram_clk_b	BRAM	O	'0'	Port B BRAM clock. Connected to AClk with same frequency, same phase alignment.
bram_en_b	BRAM	O	Zeros	BRAM Port B (read port) enable signal. Active High. Only used when C_SINGLE_PORT_BRAM = 0.

Table 2-2: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
bram_we_b [(c_s_axi_data_width/8) + c_ecc - 1:0][(2)]	BRAM	O	Zeros	BRAM Port B (read port) write enable signal. Active High. Fixed to default zeros value (port B is a read-only port to block RAM). Size is byte-wise to width of block RAM. If ECC is enabled, the BRAM_WE_A width increases to hold the ECC bits as shown in Table 3-4, page 62.
bram_addr_b [c_bram_addr_width-1:0]	BRAM	O	Zeros	BRAM Port B (read port) address bus. Bus is sized according to data width and based on C_S_AXI_BASEADDR and C_S_AXI_HIGHADDR settings. Only used when C_SINGLE_PORT_BRAM = 0.
bram_rddata_b [(8 * c_ecc + c_s_axi_data_width)-1:0]	BRAM	I		BRAM Port B (read port) read data bus. Size of BRAM read data width is equal to size of AXI slave port connection to the AXI BRAM Controller. If ECC is enabled, the BRAM_RdData_B width increases to hold the ECC bits as shown in Table 3-4, page 62. Only used when C_SINGLE_PORT_BRAM = 0.
bram_wrdata_b [(8 * c_ecc+ c_s_axi_data_width-1:0][(2)]	BRAM	O	Zeros	BRAM Port B (read port) write data bus. Fixed to default zeros value (port B is a read-only port to block RAM). Size of BRAM write data width is equal to size of AXI slave port connection to the AXI BRAM Controller. If ECC is enabled, the BRAM_WrData_B width increases to match the data width with ECC bits as shown in Table 3-4, page 62.

Notes:

1. This AXI IP slave core does not include support for the low power interface signals of the AXI-defined specification.
2. bram_we_b and bram_wrdata_b are unused port signals in the AXI BRAM Controller IP core. These signals are included in the BRAM interface for tool compatibility (and the default values are driven properly).

Register Space

AXI4-Lite ECC Registers

Table 2-3 summarizes the ECC register set that is available when ECC is enabled (C_ECC = 1) on the AXI-Lite control interface. Additional details on each register is provided in ECC Register Details, page 24.

Table 2-3: **AXI BRAM Interface Register Address Map**[(1)]

Base Address + Offset (hex)	Register	Access Type	Description
C_S_AXI_CTRL_BASEADDR + 0x0	ECC_STATUS	R/WC	ECC Status Register
C_S_AXI_CTRL_BASEADDR + 0x4	ECC_EN_IRQ	R/W	ECC Enable Interrupt Register
C_S_AXI_CTRL_BASEADDR + 0x8	ECC_ON_OFF	R/W	ECC On/Off Register
C_S_AXI_CTRL_BASEADDR + 0xC	CE_CNT	R/W	Correctable Error Counter Register
C_S_AXI_CTRL_BASEADDR + 0x100 (1)	CE_FFD [31:0]	R	Correctable Error First Failing Data Register, bits [31:0] *Unused in core. Reserved for future implementation.*
C_S_AXI_CTRL_BASEADDR + 0x104	CE_FFD [63:32]	R	Correctable Error First Failing Data Register, bits [63:32] *Unused in core. Reserved for future implementation.*
C_S_AXI_CTRL_BASEADDR + 0x108	CE_FFD [95:64]	R	Correctable Error First Failing Data Register, bits [95:64] *Unused in core. Reserved for future implementation.*
C_S_AXI_CTRL_BASEADDR + 0x10C	CE_FFD [127:96]	R	Correctable Error First Failing Data Register, bits [127:96] *Unused in core. Reserved for future implementation.*
(0x120 - 0x17C) Reserved for Expansion greater than 128-bit BRAM with ECC.			
C_BASEADDR + 0x180	CE_FFE	R	Correctable Error First Failing ECC Register *Unused in core. Reserved for future implementation.*
(0x184 - 0x1BC) Reserved for Expansion greater than 128-bit BRAM with ECC.			
C_S_AXI_CTRL_BASEADDR + 0x1C0	CE_FFA [31:0]	R	Correctable Error First Failing Address Register, bits [31:0]
C_S_AXI_CTRL_BASEADDR + 0x1C4	CE_FFA [63:32]	R	Correctable Error First Failing Address Register, bits [63:32]
C_S_AXI_CTRL_BASEADDR + 0x200	UE_FFD [31:0]	R	Uncorrectable Error First Failing Data Register, bits [31:0] *Unused in core. Reserved for future implementation.*
C_S_AXI_CTRL_BASEADDR + 0x204	UE_FFD [63:32]	R	Uncorrectable Error First Failing Data Register, bits [63:32] *Unused in core. Reserved for future implementation.*

Table 2-3: **AXI BRAM Interface Register Address Map[1] *(Cont'd)***

Base Address + Offset (hex)	Register	Access Type	Description
C_S_AXI_CTRL_BASEADDR + 0x208	UE_FFD [95:64]	R	Uncorrectable Error First Failing Data Register, bits [95:64] *Unused in core. Reserved for future implementation.*
C_S_AXI_CTRL_BASEADDR + 0x20C	UE_FFD [127:96]	R	Uncorrectable Error First Failing Data Register, bits [127:96] *Unused in core. Reserved for future implementation.*
(0x210 - 0x27C) Reserved for expansion greater than 128-bit BRAM with ECC.			
C_S_AXI_CTRL_BASEADDR + 0x280	UE_FFE	R	Uncorrectable Error First Failing ECC Register *Unused in core. Reserved for future implementation.*
(0x284 - 0x2BC) Reserved for expansion greater than 128-bit BRAM with ECC.			
C_S_AXI_CTRL_BASEADDR + 0x2C0	UE_FFA [31:0]	R	Uncorrectable Error First Failing Address Register, bits [31:0] *Unused in core. Reserved for future implementation.*
C_S_AXI_CTRL_BASEADDR + 0x2C4	UE_FFA [63:32]	R	Uncorrectable Error First Failing Address Register, bits [63:32] *Unused in core. Reserved for future implementation.*
C_S_AXI_CTRL_BASEADDR + 0x300	FI_D [31:0]	W	Fault Inject Data Register, bits [31:0]
C_S_AXI_CTRL_BASEADDR + 0x304	FI_D [63:32]	W	Fault Inject Data Register, bits [63:32]
C_S_AXI_CTRL_BASEADDR + 0x308	FI_D [95:64]	W	Fault Inject Data Register, bits [95:64]
C_S_AXI_CTRL_BASEADDR + 0x30C	FI_D [127:96]	W	Fault Inject Data Register, [127:96]
(0x310 - 0x37C) Reserved for expansion greater than 128-bit BRAM with ECC.			
C_S_AXI_CTRL_BASEADDR + 0x380	FI_ECC	W	Fault Inject ECC Register

Notes:

1. Grayed cells indicate registers that are not used in the core, but that are reserved for future implementation.

ECC Register Details

ECC_STATUS

This register holds information on the occurrence of correctable and uncorrectable errors. The status bits are independently set to '1' for the first occurrence of each error type. The status bits are cleared by writing a '1' to the corresponding bit position, that is, the status

bits can only be cleared to '0' and not set to '1' using a register write. The ECC Status register operates independently of the ECC Enable Interrupt register.

This register is available only when C_ECC = 1.

Table 2-4: **ECC Status Register (ECC_STATUS)**

Reserved		ECC_STATUS	
31	2	1	0

Table 2-5: **ECC Status Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
1	CE_STATUS	R/WC	0	If '1' a correctable error has occurred. Cleared when '1' is written to this bit position
0	UE_STATUS	R/WC	0	If '1' an uncorrectable error has occurred. Cleared when '1' is written to this bit position

ECC_EN_IRQ

This register determines if the value of the CE_STATUS and UE_STATUS bits of the ECC Status Register asserts the Interrupt output signal (ECC_Interrupt). If both CE_EN_IRQ and UE_EN_IRQ are set to '1' (enabled), the value of the Interrupt signal is the logical OR between the CE_STATUS and UE_STATUS bits.

This register is available only when C_ECC = 1.

Table 2-6: **ECC Interrupt Enable Register (ECC_EN_IRQ)**

Reserved		ECC_EN_IRQ	
31	2	1	0

Table 2-7: **ECC Interrupt Enable Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
1	CE_EN_IRQ	R/W	0	If '1', the value of the CE_STATUS bit of the ECC Status Register is propagated to the Interrupt signal. if '0', the value of the CE_STATUS bit of the ECC Status Register is not propagated to the Interrupt signal.
0	UE_EN_IRQ	R/W	0	If '1', the value of the UE_STATUS bit of the ECC Status Register is propagated to the Interrupt signal. if '0', the value of the UE_STATUS bit of the ECC Status Register is not propagated to the Interrupt signal.

ECC_ON_OFF

The ECC On/Off Control Register allows the application to enable or disable ECC checking. The design parameter, C_ECC_ONOFF_RESET_VALUE (default on) determines the reset value

for the enable/disable setting of ECC. This facilitates start-up operations when ECC might or might not be initialized in the BRAM block. When disabled, ECC checking is disabled for read, but ECC generation is active for write operations.

This register is not implemented if the value of C_ECC is 0. In this case, ECC checking is always off.

Table 2-8: **ECC On/Off Control Register (ECC_ON_OFF)**

Reserved		ECC_ON_OFF
31	1	0

Table 2-9: **ECC On/Off Control Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0	ECC_ON_OFF	R/W	Specified by design parameter, C_ECC_ONOFF_ RESET_VALUE	If '0', ECC checking is disabled on read operations. (ECC generation is enabled on write operations when C_ECC = 1). If '1', ECC checking is enabled on read operations. All correctable and uncorrectable error conditions are captured and status updated.

CE_CNT

This register counts the number of occurrences of correctable errors. It can be cleared or preset to any value by means of a register write. When the counter reaches its maximum value it does not wrap around, but stops incrementing and remains at the maximum value.

The width of the counter is defined by the value of the C_CE_COUNTER_WIDTH parameter. The AXI BRAM Controller has the value of the CE counter width set to 8 bits.

This counter register is only accessible when C_ECC = 1.

Table 2-10: **Correctable Error Counter Register (CE_CNT)**

Reserved		CE_CNT	
31	8	7	0

Table 2-11: **Correctable Error Counter Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(7:0)	CE_CNT	R/W	0	Registers holds number of correctable errors encountered

CE_FFA [31:0]

This register stores the address (bits [31:0]) of the first occurrence of an access with a correctable error. When the CE_STATUS bit in the ECC Status Register is cleared, the register

is re-enabled to store the address of the next correctable error. Storage of the failing address is enabled after reset.

This register is only implemented when C_ECC = 1.

Table 2-12: **Correctable Error First Failing Address Register (CE_FFA [31:0])**

CE_FFA [31:0]	
31	0

Table 2-13: **Correctable Error First Failing Address [31:0] Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	CE_FFA [31:0]	R	0	Address (bits [31:0]) of the first occurrence of a correctable error

CE_FFA [63:32]

This register is unused in this release of the AXI BRAM Controller IP core. The maximum address width setting on the block RAM is 32.

This register stores the address (bits [63:32]) of the first occurrence of an access with a correctable error. When the CE_STATUS bit in the ECC Status Register is cleared, the register is re-enabled to store the address of the next correctable error. Storage of the failing address is enabled after reset.

This register is only implemented when C_ECC = 1.

Table 2-14: **Correctable Error First Failing Address Register (CE_FFA [63:32])**

CE_FFA [63:32]	
31	0

Table 2-15: **Correctable Error First Failing Address [63:32] Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	CE_FFA [63:32]	R	0	Address (bits [63:32]) of the first occurrence of a correctable error

CE_FFD [31:0]

This register is unused in this release of the AXI BRAM Controller IP core.

CE_FFD [63:32]

This register is unused in this release of the AXI BRAM Controller IP core.

CE_FFD [95:64]

This register is unused in this release of the AXI BRAM Controller IP core.

CE_FFD [127:96]

This register is unused in this release of the AXI BRAM Controller IP core.

CE_FFE

This register is unused in this release of the AXI BRAM Controller IP core.

UE_FFA [31:0]

This register is unused in this release of the AXI BRAM Controller IP core.

UE_FFA [63:32]

This register is unused in this release of the AXI BRAM Controller IP core.

UE_FFD [31:0]

This register is unused in this release of the AXI BRAM Controller IP core.

UE_FFD [63:32]

This register is only used when the C_S_AXI_DATA_WIDTH > 32.

This register stores the (uncorrected) failing data (bits [63:32]) of the first occurrence of an access with an uncorrectable error. When the UE_STATUS bit in the ECC Status Register is cleared, this register is re-enabled to store the data of the next uncorrectable error. Storage of the failing data is enabled after reset.

This register is only implemented when C_ECC = 1.

Table 2-16: **Uncorrectable Error First Failing Data Register (UE_FFD [63:32])**

UE_FFD [63:32]	
31	0

Table 2-17: **Uncorrectable Error First Failing Data [63:32] Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	UE_FFD [63:32]	R	0	Data (bits [63:32) of the first occurrence of an uncorrectable error.

UE_FFD [95:64]

This register is only used when the C_S_AXI_DATA_WIDTH > 64.

This register is unused in this release of the AXI BRAM Controller IP core; however, it will be available in a future release of the IP core when ECC is supported for 128-bit AXI BRAM data width configurations.

This register stores the (uncorrected) failing data (bits [95:64]) of the first occurrence of an access with an uncorrectable error. When the UE_STATUS bit in the ECC Status Register is cleared, this register is re-enabled to store the data of the next uncorrectable error. Storage of the failing data is enabled after reset.

This register is only implemented when C_ECC = 1.

Table 2-18: **Uncorrectable Error First Failing Data Register (UE_FFD [95:64])**

UE_FFD [95:64]	
31	0

Table 2-19: **Uncorrectable Error First Failing Data [95:64] Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	UE_FFD [95:64]	R	0	Data (bits [95:64]) of the first occurrence of an uncorrectable error.

UE_FFD [127:96]

This register is only used when the C_S_AXI_DATA_WIDTH > 64.

Unused in this release of the AXI BRAM Controller IP core. Will be available in a future release of the IP core when ECC is supported for 128-bit AXI BRAM data width configurations.

This register stores the (uncorrected) failing data (bits [127:96]) of the first occurrence of an access with an uncorrectable error. When the UE_STATUS bit in the ECC Status Register is cleared, this register is re-enabled to store the data of the next uncorrectable error. Storage of the failing data is enabled after reset.

This register is only implemented when C_ECC = 1.

Table 2-20: **Uncorrectable Error First Failing Data Register (UE_FFD [127:96])**

UE_FFD [127:96]	
31	0

Table 2-21: **Uncorrectable Error First Failing Data [127:96] Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	UE_FFD [127:96]	R	0	Data (bits [127:96]) of the first occurrence of an uncorrectable error.

UE_FFE

This register is unused in this release of the AXI BRAM Controller IP core.

FI_D0

This register is used to inject errors in data (bits [31:0]) written to the block RAM and can be used to test the error correction and error signaling. The bits set in the register toggle the corresponding data bits (word 0 or bits [31:0]) of the subsequent data written to the block RAM without affecting the ECC bits written. After the fault has been injected, the Fault Injection Data Register is cleared automatically.

The register is only implemented if C_FAULT_INJECT = 1 and C_ECC = 1.

Injecting faults should be performed in a critical region in software; that is, writing this register and the subsequent write to AXI BRAM must not be interrupted.

Table 2-22: **Fault Injection Data Register (FI_D0)**

FI_D0	
31	0

Table 2-23: **Fault Injection Data (Word 0) Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	FI_D0	W	0	Bit positions set to '1' toggle the corresponding bits [31:0] of the next data word written to the block RAM. The register is automatically cleared after the fault has been injected.

For larger data width BRAMs with ECC (64- and 128-bit configurations), special consideration must be given across FI_D0, FI_D1, FI_D2 and FI_D3 such that only a single error condition is introduced.

FI_D1

This register is only used when the C_S_AXI_DATA_WIDTH > 32.

This register is used to inject errors in data (bits [63:32]) written to the block RAM and can be used to test the error correction and error signaling. The bits set in the register toggle the corresponding data bits (word 1 or bits [63:32]) of the subsequent data written to the

block RAM without affecting the ECC bits written. After the fault has been injected, the Fault Injection Data Register is cleared automatically.

This register is only implemented if C_FAULT_INJECT = 1 and C_ECC = 1.

Injecting faults should be performed in a critical region in software; that is, writing this register and the subsequent write to AXI BRAM must not be interrupted.

Table 2-24: **Fault Injection Data Register (FI_D1)**

FI_D1	
31	0

Table 2-25: **Fault Injection Data (Word 1) Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	FI_D1	W	0	Bit positions set to '1' toggle the corresponding bits [63:32] of the next data word written to the block RAM. The register is automatically cleared after the fault has been injected.

FI_D2

This register is only used when the C_S_AXI_DATA_WIDTH > 64.

This register is used to inject errors in data (bits [95:64]) written to the block RAM and can be used to test the error correction and error signaling. The bits set in the register toggle the corresponding data bits (word 2 or bits [95:64]) of the subsequent data written to the block RAM without affecting the ECC bits written. After the fault has been injected, the Fault Injection Data Register is cleared automatically.

This register is only implemented if C_FAULT_INJECT = 1 and C_ECC = 1.

Injecting faults should be performed in a critical region in software; that is, writing this register and the subsequent write to AXI BRAM must not be interrupted.

Table 2-26: **Fault Injection Data Register (FI_D2)**

FI_D2	
31	0

Table 2-27: **Fault Injection Data (Word 2) Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	FI_D2	W	0	Bit positions set to '1' toggle the corresponding bits [95:64] of the next data word written to the block RAM. The register is automatically cleared after the fault has been injected.

FI_D3

This register is only used when the C_S_AXI_DATA_WIDTH > 64.

This register is used to inject errors in data (bits [127:96]) written to the block RAM and can be used to test the error correction and error signaling. The bits set in the register toggle the corresponding data bits (word 3 or bits [127:96]) of the subsequent data written to the block RAM without affecting the ECC bits written. After the fault has been injected, the Fault Injection Data Register is cleared automatically.

This register is only implemented if C_FAULT_INJECT = 1 and C_ECC = 1.

Injecting faults should be performed in a critical region in software; that is, writing this register and the subsequent write to AXI BRAM must not be interrupted.

Table 2-28: **Fault Injection Data Register (FI_D3)**

FI_D3	
31	0

Table 2-29: **Fault Injection Data (Word 3) Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 31	FI_D3	W	0	Bit positions set to '1' toggle the corresponding bits [127:96] of the next data word written to the block RAM. The register is automatically cleared after the fault has been injected.

FI_ECC

This register is used to inject errors in the generated ECC written to the block RAM and can be used to test the error correction and error signaling. The bits set in the register toggle the corresponding ECC bits of the next data written to block RAM. After the fault has been injected, the Fault Injection ECC Register is cleared automatically.

The register is only implemented if C_FAULT_INJECT = 1 and C_ECC = 1.

Injecting faults should be performed in a critical region in software; that is, writing this register and the subsequent write to AXI BRAM must not be interrupted.

Table 2-30 and Table 2-31 describes the register bit usage when C_S_AXI_DATA_WIDTH = 32.

Table 2-30: **Fault Injection ECC Register (FI_ECC) for 32-bit BRAM**

Reserved		FI_ECC	
31	7	6	0

Table 2-31: **Fault Injection ECC Register Bit Definitions for 32-bit BRAM**

Bit(s)	Name	Core Access	Reset Value	Description
(6:0)	FI_ECC	R	0	Bit positions set to '1' toggle the corresponding bit of the next ECC written to the block RAM. The register is automatically cleared after the fault has been injected.

Table 2-32 and Table 2-33 describe the register bit usage when C_S_AXI_DATA_WIDTH = 64.

Table 2-32: **Fault Injection ECC Register (FI_ECC) for 64-bit BRAM**

Reserved		FI_ECC	
31	8	7	0

Table 2-33: **Fault Injection ECC Register Bit Definitions for 64-bit BRAM**

Bit(s)	Name	Core Access	Reset Value	Description
(7:0)	FI_ECC	R	0	Bit positions set to '1' toggle the corresponding bit of the next ECC written to the block RAM. The register is automatically cleared after the fault has been injected.

Table 2-34 and Table 2-35 describe the register bit usage when C_S_AXI_DATA_WIDTH = 128.

Table 2-34: **Fault Injection ECC Register (FI_ECC) for 128-bit BRAM**

Reserved		FI_ECC	
31	9	8	0

Table 2-35: **Fault Injection ECC Register Bit Definitions for 128-bit BRAM**

Bit(s)	Name	Core Access	Reset Value	Description
8	FI_ECC [8]	R	0	Bit position set to '1' toggle the corresponding bit of the next ECC written to the block RAM. The register is automatically cleared after the fault has been injected.
(7:0)	FI_ECC [7:0]	R	0	Bit positions set to '1' toggle the corresponding bit of the next ECC written to the block RAM. The register is automatically cleared after the fault has been injected.

Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the core.

General Design Guidelines

AXI4-Lite Controller Design

The AXI BRAM Controller IP core can be configured in a simplified footprint core by setting the design parameter, C_S_AXI_PROTOCOL = AXI4LITE.

In this configuration, the AXI BRAM Controller can be configured to interface to a single port or dual port BRAM. All arbitration to the block RAM (in a single port mode) is performed on the AXI write address (AW) and read address (AR) channels. Only one active operation is allowable in the core at a time. The AXI BRAM Controller AW and AR channel interfaces assign priority to the read channel when presented with simultaneous assertions of the ARVALID and AWVALID signals; ensuring the AXI BRAM Controller is compliant with any AXI4-Lite master. When the AXI BRAM Controller is created in a system topology with the AXI Interconnect IP, it is ensured that ARVALID and AWVALID are not asserted at the same time and the AXI Interconnect handles the arbitration between the write and read channels. The AXI Interconnect provides optimizations in the AXI BRAM Controller core through the build process by duplicating the ARADDR and AWADDR signals.

To configure the dual-port BRAM mode to the BRAM block, ensure the design parameter, C_SINGLE_PORT_BRAM = 0.

The AXI4-Lite BRAM Controller is designed for minimal FPGA resource utilization. Minimal registers are utilized, only those to support the register outputs to the AXI4-Lite interface according to standard.

AXI4-Lite Single Write

The write datapath timing in the core is shown in Figure 3-1 for the AXI4-Lite interface connections. Figure 3-1 assumes that no valid transaction is active on the AXI AR and R channels.

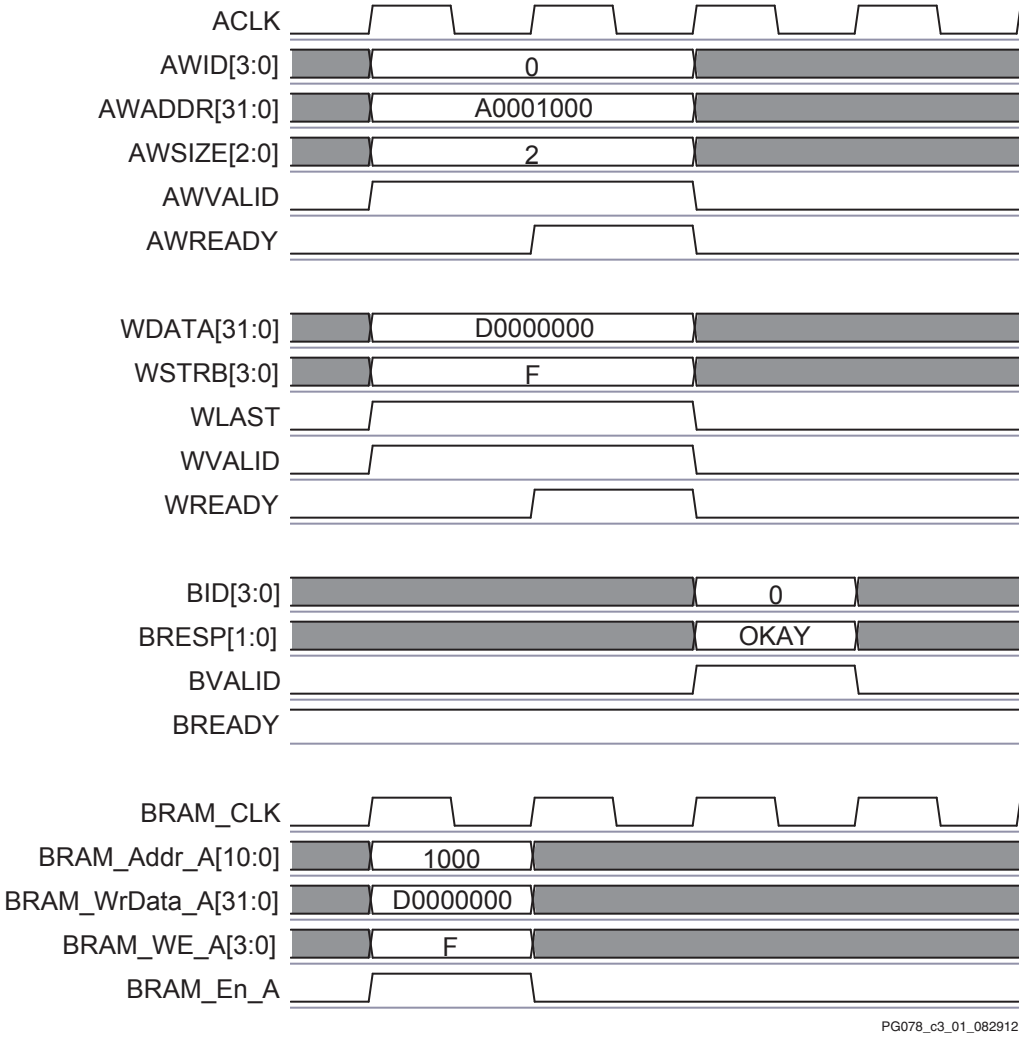

Figure 3-1: **AXI4-Lite Write Timing**

AXI4-Lite Multiple Write

The AXI4-Lite BRAM Controller is able to accept continuous single write operations as shown in Figure 3-2.

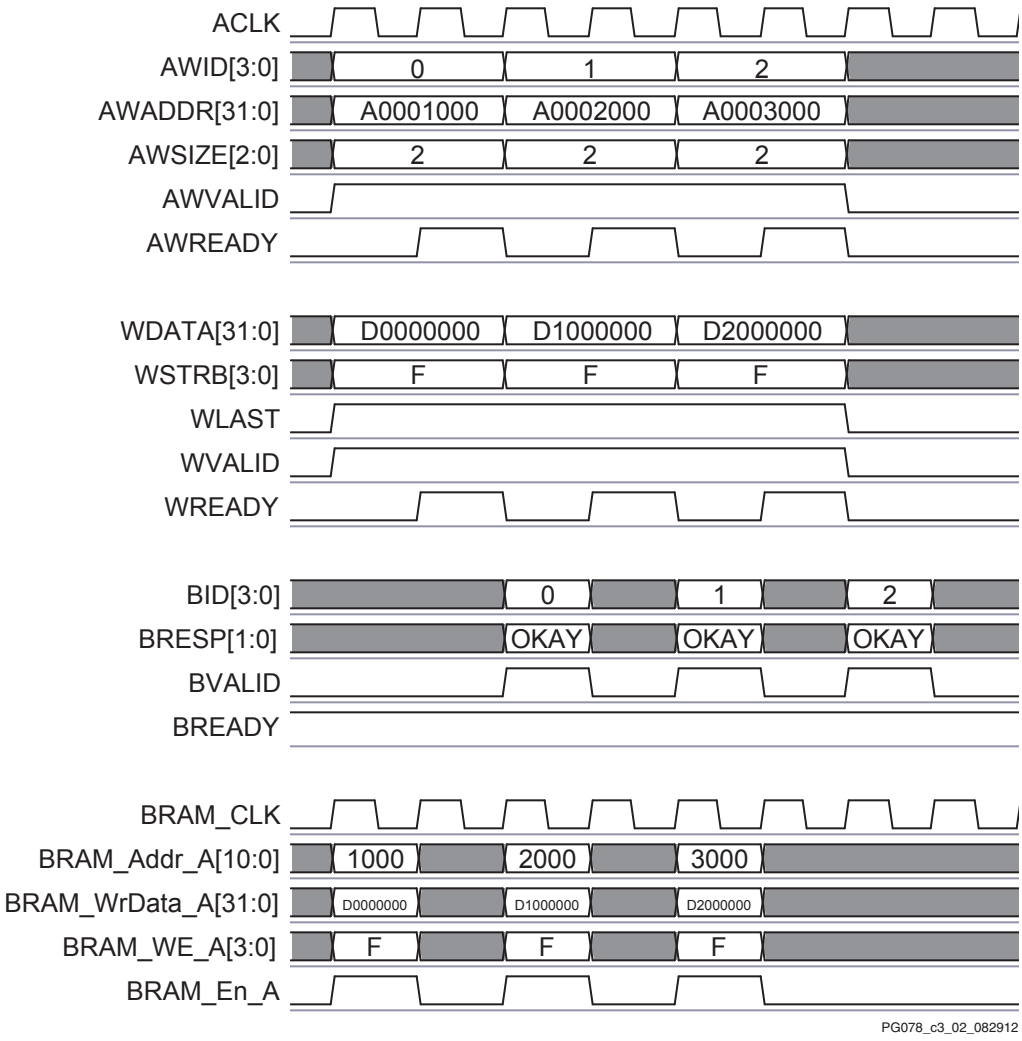

Figure 3-2: **Multiple AXI4-Lite Write Transactions**

AXI4-Lite Single Read

The read datapath timing is shown in Figure 3-3 for AXI4-Lite transactions. Multiple read request operations illustrate the timing of the controller and on the BRAM interface.

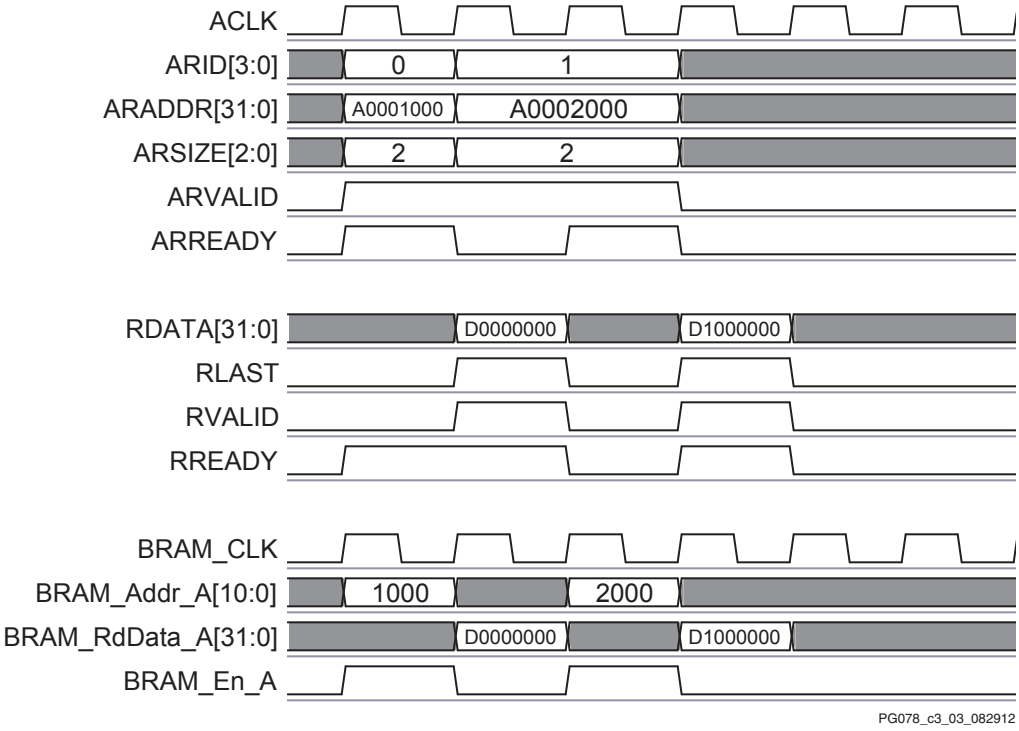

Figure 3-3: **AXI4-Lite Read Timing**

AXI4-Lite Write with ECC

Figure 3-4 illustrates the timing of the read-modify-write (RMW) sequence when C_ECC = 1 for write transfers on the AXI4-Lite interface. To save resources the AWADDR and WDATA are not registered in the AXI4-Lite BRAM controller, so the AWREADY and WREADY output flag assertions are delayed until the RMW sequence is completed. Also depicted is the capability of the AXI4-Lite controller to handle back-to-back write request operations.

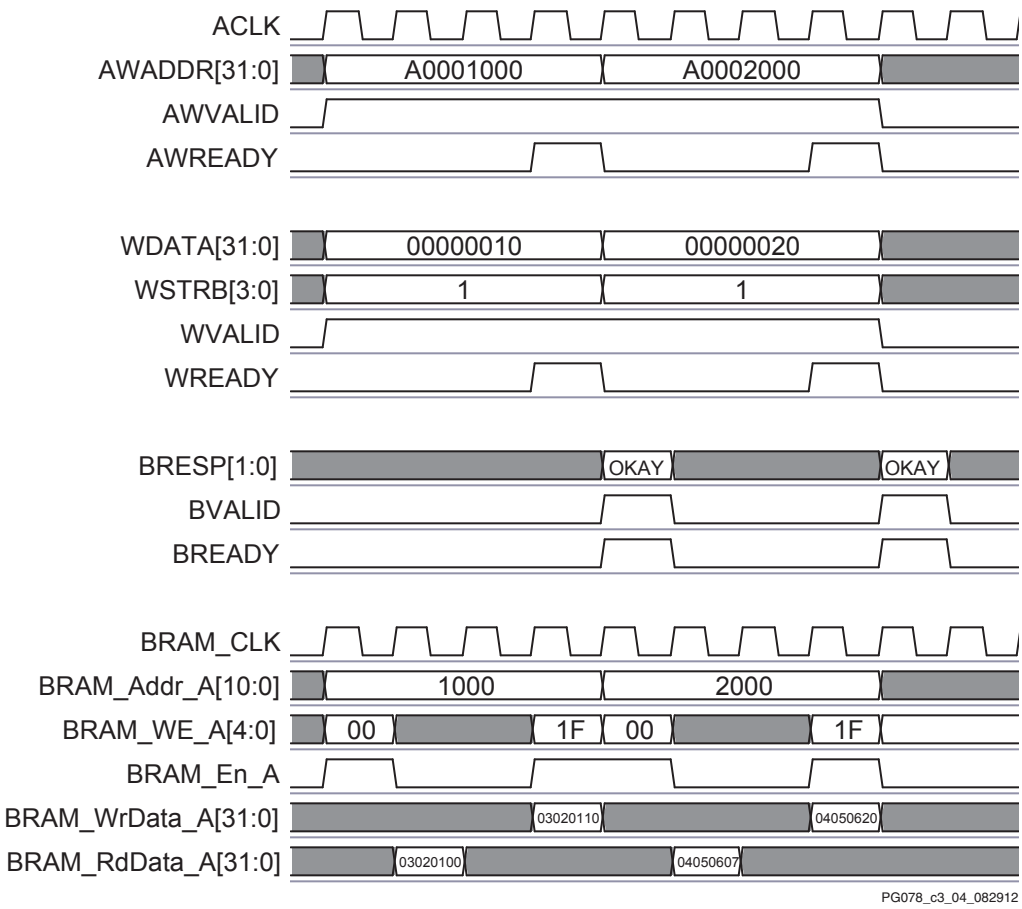

Figure 3-4: **AXI4-Lite Write with ECC**

AXI4 Controller Design

The AXI BRAM Controller meets the performance requirement of AXI based systems and provides minimal latency to/from block RAM. Single clock cycle latency is achieved on each channel Valid to Ready response, depending on the current core activity. The AXI BRAM Controller separates the write and read channel activity (when C_SINGLE_PORT_BRAM = 0), so that the controller can handle simultaneous read and write operations from AXI. All timing relationships between the write address and write data channel (as well as read address and read data channels) are designed to meet the AXI standard, so as to avoid any deadlock situations.

No address decoding is performed by the AXI BRAM Controller; all operations received are accepted. The Xilinx AXI Interconnect provides the address decode.

All AXI master size matching to the data width of the block RAM is handled by the Xilinx AXI Interconnect module. The slave IP port of the AXI BRAM Controller on the AXI Interconnect is equal to the width of the block RAM (equal to the size of the AXI BRAM Controller). The AXI BRAM Controller does not need to know the data width or burst size of the requesting master, as the Interconnect translates all operations to fit to the data width and burst size of the data bus connected to the AXI BRAM Controller (and block RAM).

AXI Singles

A single AXI transaction is qualified by the AWSIZE/ARSIZE equal to "000" and AWLEN/ARLEN set to "0000" for 1 data transfer. All AXI4-Lite operations are supported by the AXI BRAM Controller when (C_S_AXI_PROTOCOL = AXI4).

Bursting

All burst types presented to the AXI BRAM Controller are supported. However, fixed burst types are translated into incrementing burst types issued to the block RAM. Wrapping burst types are support for processor cacheline read and writes with block RAM.

Each burst is specified to have N data transfers (1-256 specified in AWLEN/ARLEN) of M bytes (1-128 bytes specified in AWSIZE/ARSIZE). The AXI BRAM Controller supports up to the AXI4 extension on burst sizes to 256 data beats.

The size of the burst must be equal or less than the size of the BRAM data width. For instance, 64-bit BRAM only allows bursts of up to 64 byte sizes. Burst sizes less than the full width of the AXI BRAM Controller data bus is referred to as a "narrow" burst. All "narrow" bursts are supported by the AXI BRAM Controller. For example, a master is requesting a byte burst operation to/from a 64-bit data wide BRAM. For "narrow" bursts, the AXI protocol defines that the valid byte lanes rotate through the correct byte lanes. In the AXI BRAM Controller, each write data beat of a burst on the AXI is translated to a write operation to the block RAM. The BRAM byte enables are configured such that only the valid bytes are stored in memory. The AXI BRAM Controller does not buffer any subsize data beats into the full width to the block RAM. An example is shown in the section, Narrow Write Bursting, page 46.

Unaligned burst transfers are allowed and the AXI master must indicate this with the lower order bits of the address bus provided with the write address channel data handshaking. An example is shown in section, Unaligned Write Bursting, page 47. Unaligned burst transfers can also be indicated with an aligned address, but use the write data channel data strobes to indicate the valid byte lanes.

Each write and read channel of the AXI BRAM Controller utilizes an address counter that is loaded at the beginning of the burst operation. AXI provides the starting address of the burst, and the AXI BRAM Controller increments the address based on the BRAM data width.

AXI does not allow any read nor write burst termination. Each AXI master must complete each burst transaction that is initiated. Each burst transaction is complete when the LAST signal is asserted, by the master of writes, and by the AXI BRAM Controller on reads.

The write enables presented to the block RAM are calculated based on burst type, address offset of the write address, along with the AXI write data strobes. The write enables are generated on a per byte basis to the block RAM. Write enables are 4 bits wide for 32-bit BRAMs, 8 bits wide for 64-bit BRAMs, and 16 bits wide for 128-bit BRAM instantiations.

Cacheline

Cacheline operations are implemented as WRAP burst types on the AXI bus when presented to the block RAM. The allowable burst sizes for WRAP bursts are 2, 4, 8, and 16 data transfers. The AWBURST/ARBURST must be set to “10” for the WRAP burst type.

WRAP bursts are handled in the address generator logic of the controller to the block RAM. The address seen by the block RAM must increment to the boundary, then wrap back around to the beginning of the cacheline address. For example, a processor issuing a target word first cache line read request to address, 0x04h. A 32-bit BRAM sees the following sequence of address-requested reads: 0x04h, 0x08h, 0x0Ch, 0x00h. This example is illustrated in Cacheline Reads, page 56 for a cacheline read.

Pipelining

Each write and read channel interface to the AXI can hold two active addresses/operations. The AXI BRAM Controller IP allows two active write and two active read transactions to be captured and held (when C_SINGLE_PORT_BRAM = 0). The data provided with each address channel operation must remain in order. No out of order operations are allowed in the AXI BRAM Controller. The pipelining capability allows the same master or another master to request a subsequent write or read transaction. With the storage of multiple write addresses in the AXI BRAM Controller, the write data may not be immediately captured and stored by the controller. The write data channel interface of the AXI BRAM Controller does not assert WREADY until the block RAM is ready to accept the data of the pipelined write transaction. On the write address channel interface, the AXI BRAM Controller keeps AWREADY asserted, until the pipeline full condition is reached. At this point, the AWREADY is negated and no further write addresses are accepted, until the first address data phase is complete and the second pipelined address can be processed by the controller. An example of the timing relationship is shown in Write Pipeline, page 49.

On read transactions, two deep address pipelining is supported in the AXI BRAM Controller. The AXI BRAM Controller issues the request to block RAM and provides data on the read data channel, as long as the requesting master asserts RREADY.

When the IP core is configured for single port BRAM utilization (C_SINGLE_PORT_BRAM = 1), a 2-deep address pipeline exists in the core, but is shared between requesting AR and AW address transactions.

ID Wrapping

The AXI BRAM Controller does not support data reordering. All ID values are wrapped back around on the data channel during handshaking. The AWID (on the write address channel) is captured and returned as the BID signal (of the write response channel) during the write data transaction. The ARID (on the read address channel) is captured and returned as the RID signal (of the read response channel) during the read data response transaction.

Power Considerations

To conserve power consumption on the BRAM interface, the BRAM enable signal is only asserted during active read or write operations. The enable signal on each port to block RAM is negated when no activity exists on the AXI bus.

ECC

When C_ECC = 1, the AXI BRAM Controller implements a read-modify-write on all write transfers. Read operation timing remains identical to transfers when ECC is disabled. The read-modify-write latency must be accounted for on all write transfers to block RAM.

AXI4 Timing

The timing diagrams shown in the subsequent sections represent the timing relationships of the AXI slave AXI BRAM Controller IP connection to the Xilinx AXI Interconnect. All write operations are initiated on the Write Address Channel (AW) of the AXI bus which specifies the type of write transaction and the corresponding address information. The handshaking protocol follows a Valid and Ready mechanism. All address and control information is only valid when the Valid signal is asserted. When the slave asserts the Ready signal, it captures the signals and accepts the operation. The Write Data Channel (W) communicates all write data for the single or burst write operation. The Write Response Channel (B) is used as the handshaking or response on the write operation.

On read operations, the Read Address Channel (AR) communicates all address and control information when the AXI master asserts the Valid signal. The slave IP (or AXI BRAM Controller) asserts the Ready signal on the RAC when the read operation can be processed. When the read data is available to send back to the AXI master, the Read Data Channel (R) translates the data and status of the operation.

AXI4 Operations

Single Write

Figure 3-5 illustrates an example for the timing of an AXI single 32-bit write operation to a 32-bit wide BRAM. This example illustrates the single write to BRAM address 0x1000h,

provided that C_S_AXI_BASEADDR is set to 0xA000 0000 and the C_S_AXI_HIGHADDR allows space for more than 4k of addressable BRAM.

As recommended, the AXI BRAM Controller keeps the AWREADY signal asserted on the bus, as the address can be captured in the clock cycle when the S_AXI_AWVALID and S_AXI_AWREADY signals are both asserted. Once the write address pipeline (two deep) is full, then the slave AXI BRAM Controller negates the AWREADY registered output signal.

The same principle applies to the write data channel, WVALID and WREADY signals. The AXI BRAM Controller negates the WREADY signal when the data pipeline writing to block RAM is full. This condition may occur when the AXI BRAM Controller is processing a prior burst write data operation.

When ECC is enabled on full data width BRAM write transfers, the timing of the transaction is identical to when C_ECC = 0.

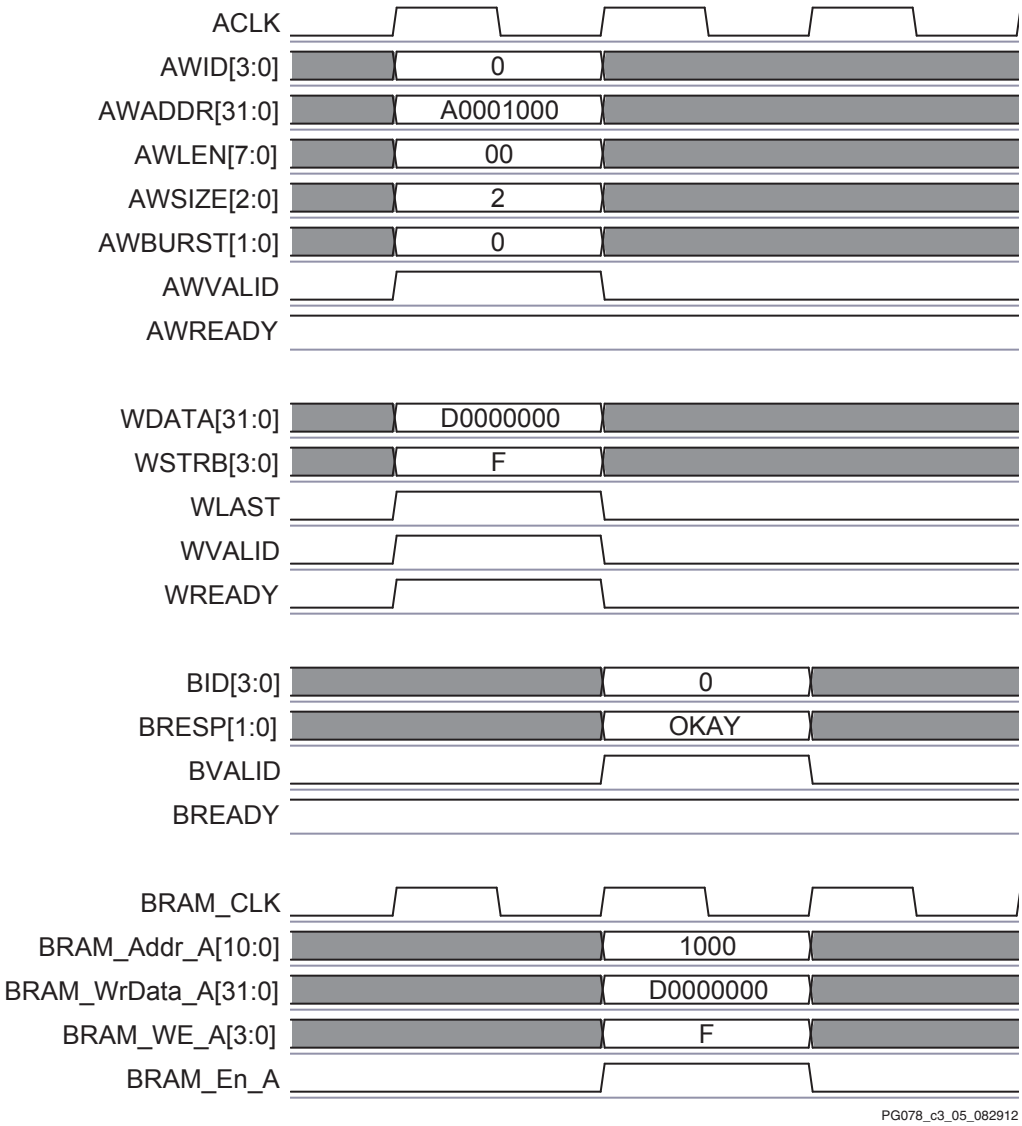

Figure 3-5: **AXI Single Write Timing Diagram**

It is possible on the write data channel for the data to be presented to the AXI BRAM Controller prior to the write address channel (AWVALID). In this case, the AXI BRAM Controller does not initiate the write transactions (the write data is ignored), until the write address channel has valid information for the AXI BRAM Controller to accept.

Single Read

Figure 3-6 illustrates an example of the timing for an AXI single read operation from a 32-bit BRAM.

The registered ARREADY signal output on the AXI Read Address Channel interface defaults to a high assertion. The AXI BRAM Controller can accept the read address in the same clock cycle as the ARVALID signal is first valid. The AXI BRAM Controller registers in the read

address when the ARVALID and the ARREADY signals are both asserted. When the AXI BRAM Controller read address pipeline is full (two-deep), it clears the ARREADY signal until the pipeline is in a non full condition.

The AXI BRAM Controller can accept the same clock cycle assertion of the RREADY by the master, if the master can accept data immediately. When the RREADY signal is asserted on the AXI bus by the master, the AXI BRAM Controller negates the RVALID signal.

For single read transactions, the RLAST is asserted with the RVALID by the AXI BRAM Controller.

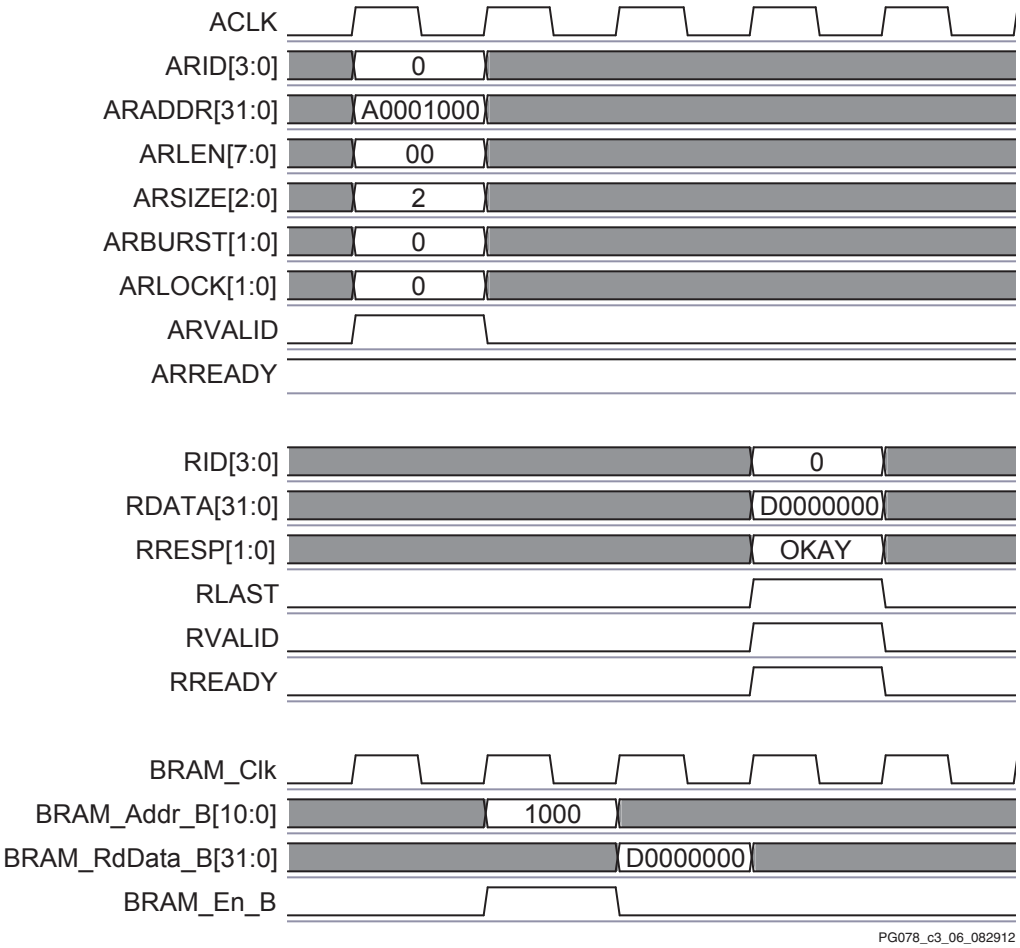

Figure 3-6: **AXI Single Read Timing Diagram**

On AXI read transactions, the read data always follows the read address handshaking. The AXI BRAM Controller does not assert RVALID until both the ARVALID and ARREADY signals are asserted in the same clock cycle. In other words, there is no ability for early access to the AXI BRAM Controller nor internal caching ability in the AXI BRAM Controller.

AXI Burst Operations

Write Burst

Figure 3-7 illustrates an example of the timing for an AXI write burst of four words to a 32-bit BRAM. The address write channel handshaking stage communicates the burst type as INCR, the burst length of 4 data transfers (AWLEN = 0011b). The write burst utilizes all byte lanes of the AXI data bus to the block RAM (AWSIZE = 010b). The write burst shown in Figure 3-7 is set to start at BRAM address 0x1000h, provided that the C_S_AXI_BASEADDR design parameter is set to 0xA000 0000 and the C_S_AXI_HIGHADDR allows space for more than 4k of addressable block RAM.

On the AXI write transactions, the slave does not wait for the write data channel, WVALID signal to be asserted prior to the assertion of the write address channel signal, AWREADY. This could potentially cause a deadlock condition and is not allowed.

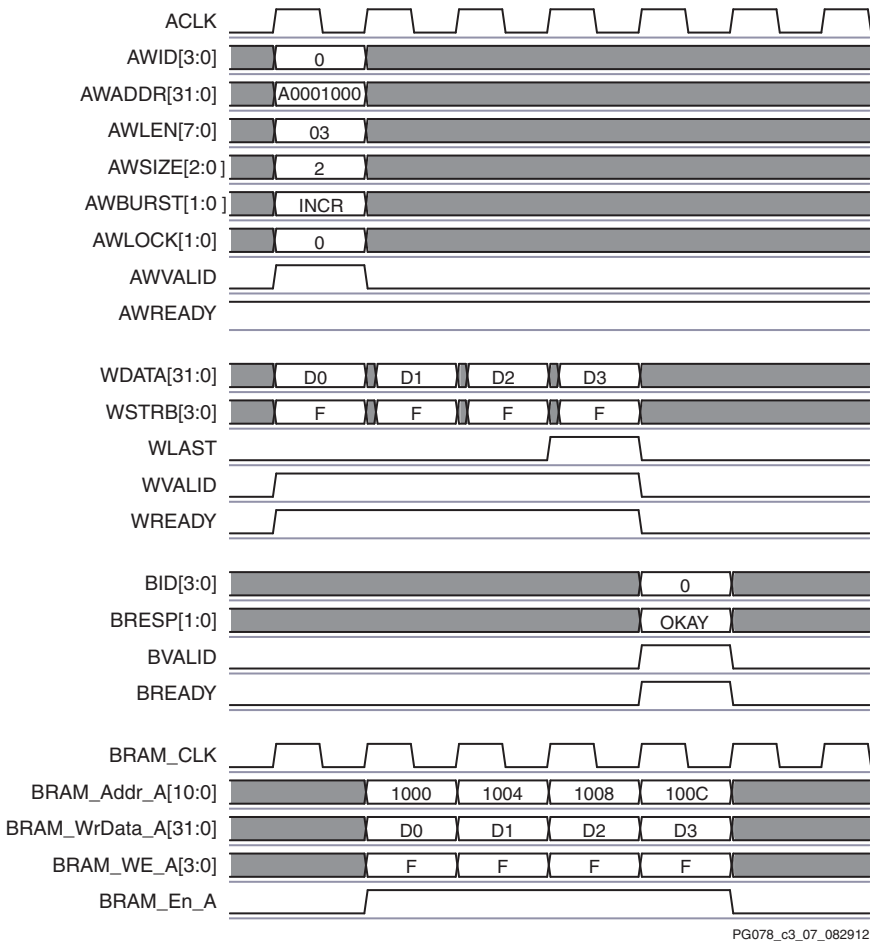

Figure 3-7: **AXI Burst Write Timing Diagram**

Narrow Write Bursting

Figure 3-8 illustrates an example of the AXI BRAM Controller supporting a *narrow* burst operation. A *narrow* burst is defined as a master bursting a data size smaller than the BRAM data width. If the burst type (AWBURST) is set to INCR or WRAP, then the valid data on the BRAM interface to the AXI bus rotates for each data beat. The AXI BRAM Controller handles each data beat on the AXI as a corresponding data beat to the block RAM, regardless of the smaller valid byte lanes. In this scenario, the AXI WSTRB is translated to the BRAM write enable signals. The BRAM address only increments when the full address (data) width boundary is met with the *narrow* write to block RAM.

Figure 3-8 illustrates an example of AXI *narrow* bursting with a 32-bit BRAM and the AXI master request is a halfword burst of 4 data beats. AWSIZE is set to 001b.

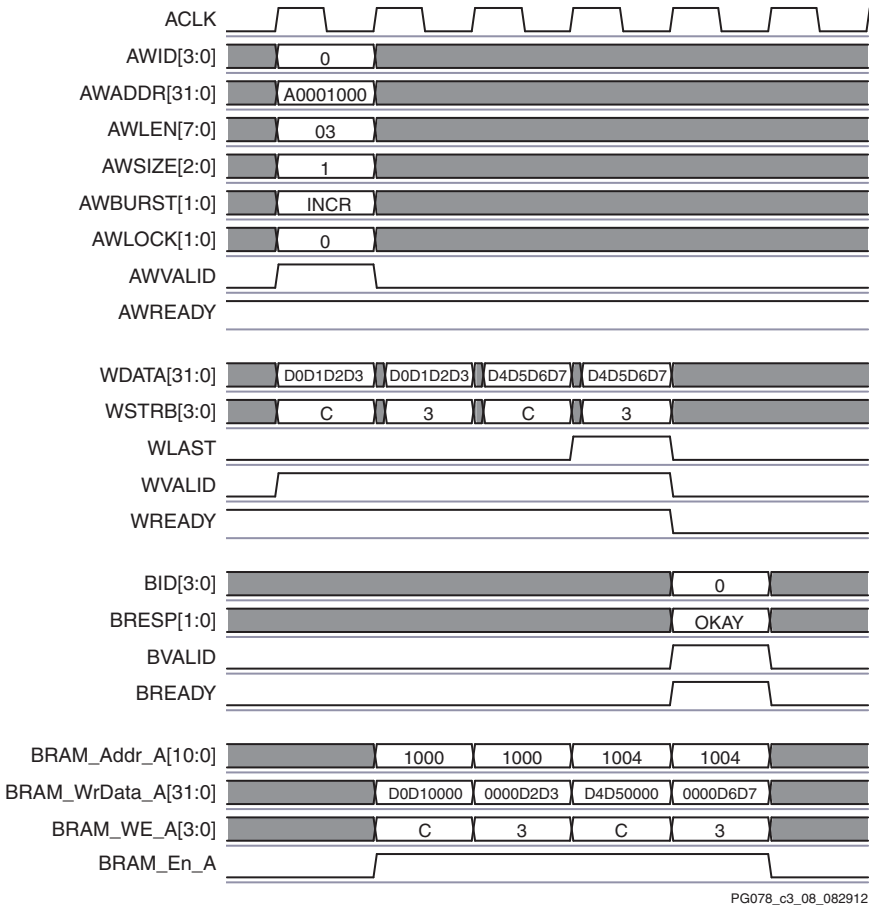

Figure 3-8: **AXI Narrow Burst Write Diagram**

Unaligned Write Bursting

The AXI BRAM Controller supports unaligned burst transfers. Unaligned burst transfers for example, occur when a 32-bit word burst size does not start on an address boundary that matches a word memory location. The starting memory address is permitted to be something other than 0x0h, 0x4h, 0x8h, etc.

The example shown in Figure 3-9, illustrates an unaligned word burst transaction of 4 data beats, that starts at address offset, 0x1002h (provided that `C_S_AXI_BASEADDR` is set to 0xA000 0000 and `C_S_AXI_HIGHADDR` allows more than 4k of addressable memory). The associated timing relationship is shown is Figure 3-10.

TIP: *Note the unaligned address corresponds to the BRAM_WE signals on the write port to reflect the valid byte lanes.*

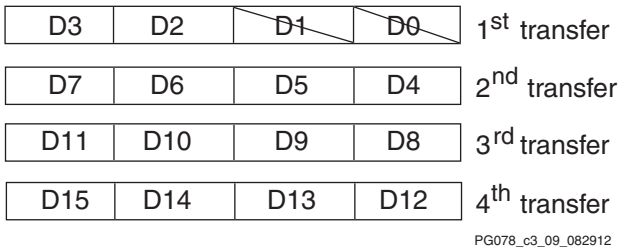

Figure 3-9: **AXI Unaligned Burst Write Transfers**

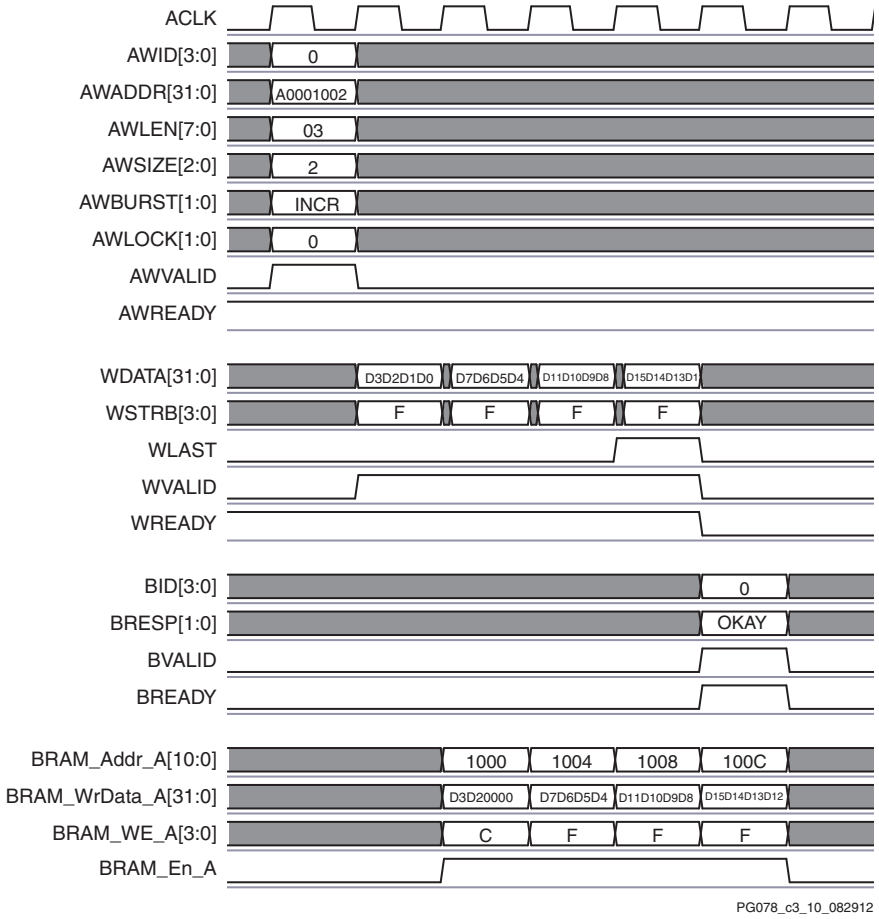

Figure 3-10: **AXI Unaligned Burst Write Timing**

ECC Write Burst

Figure 3-11 illustrates the timing for an AXI write burst of two data beats.

TIP: *Note the additional latency due to the read-modify-write sequence required to correctly update the ECC check bits in block RAM.*

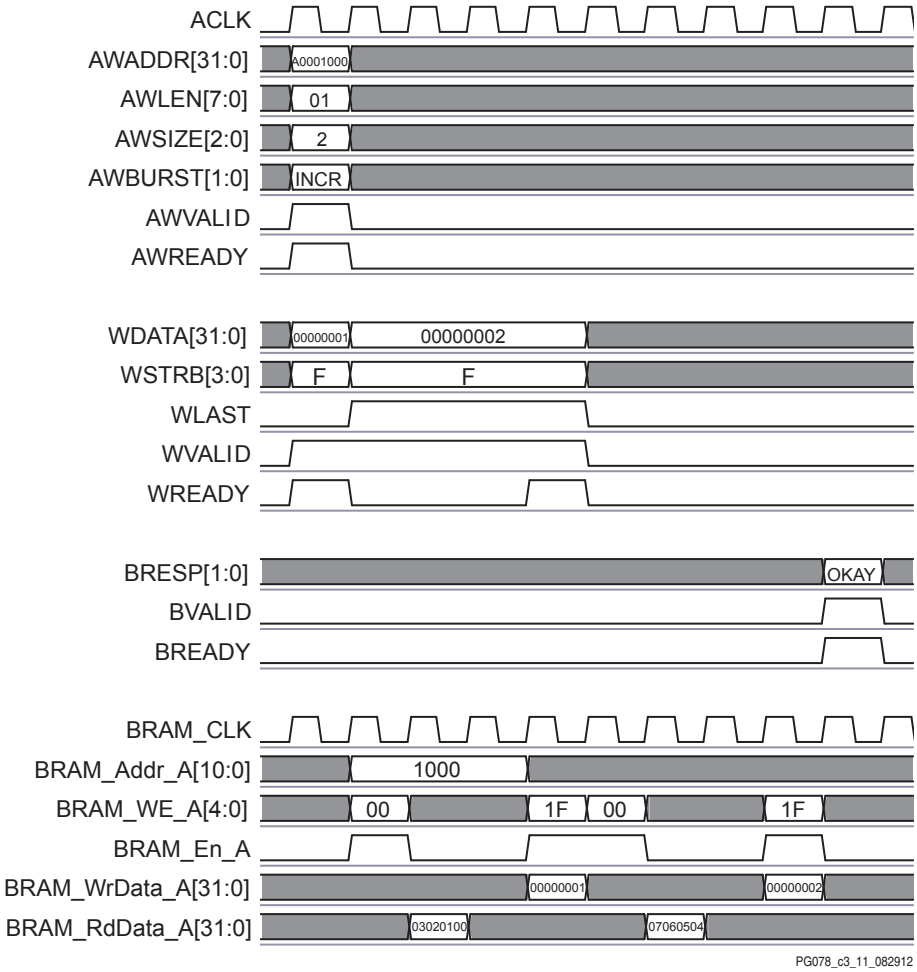

Figure 3-11: **AXI Write Burst w/ ECC**

Write Pipeline

The AXI BRAM Controller IP core is capable of supporting up to two active write operations. Once the write address pipeline is full, the controller negates `AWREADY` to the bus. Only when the first pipelined operation is complete, may a new write transfer occur subsequently (indicated by the assertion of `AWREADY`).

The AXI BRAM Controller does not support write data interleaving, so the data on the write data channel must be in the same order as the addresses (for that data) presented on the write address channel.

The write data channel responds to the write transfer with the assertion of `WREADY`. The AXI BRAM Controller accepts data when the write data channel is not already busy from a previous transaction. The AXI BRAM Controller supports the early assertion of `WREADY` to `WVALID` when asserted prior to `AWVALID` and `AWREADY`. The AXI BRAM Controller only accepts one data transfer before the write address is accepted.

The AXI BRAM Controller can accept data on the write data channel in the same clock cycle when the write address is valid and accepted by the controller. The WREADY signal remains asserted to continually accept the burst, as the BRAM address counter is loaded and the write data can pass to block RAM.

The AXI BRAM Controller ensures that BREADY is not asserted prior to WVALID and WREADY.

Figure 3-12 illustrates the capability to pipeline write addresses in the AXI BRAM Controller. This example illustrates when a gap is seen on the write data channel by the AXI BRAM Controller from the AXI Interconnect.

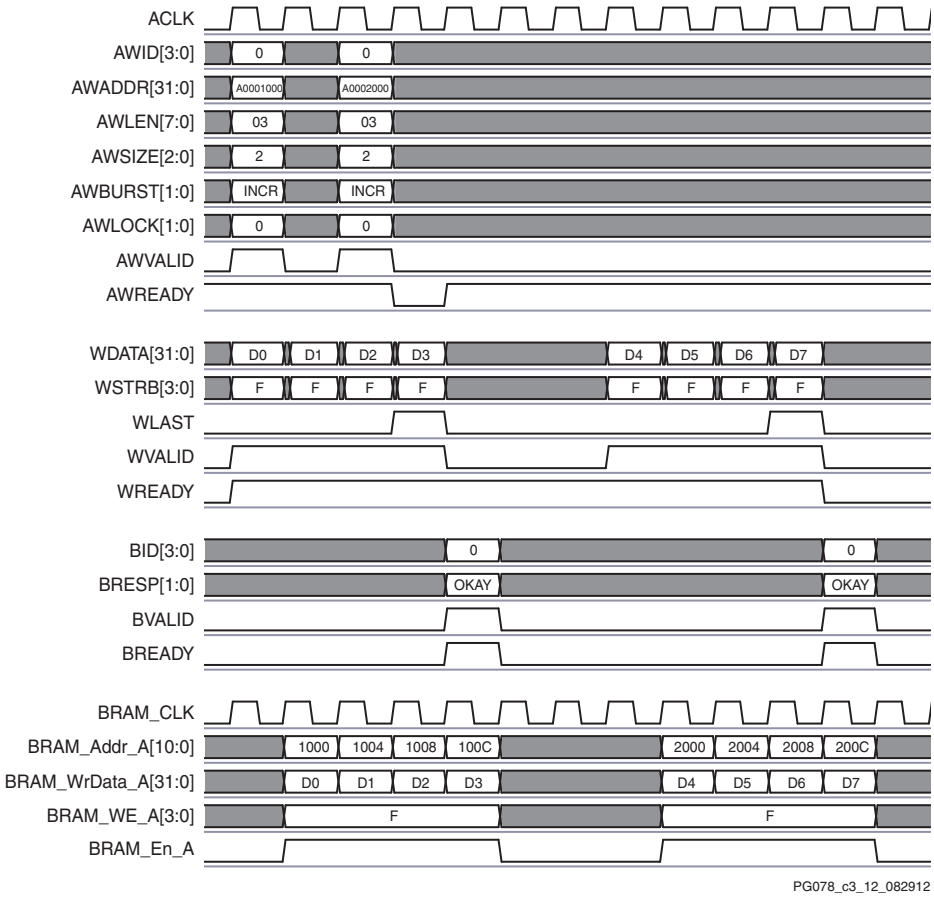

Figure 3-12: **AXI Pipelined Burst Write Transfers**

The AXI BRAM Controller can support back-to-back write burst operations if supplied with a continuous data stream from the AXI Interconnect. In this case, there are no idle clock cycles on the BRAM interface between the two pipelined write burst operations. Figure 3-13 illustrates the timing for back-to-back pipelined write bursts of four data beats.

To achieve 100% BRAM interface utilization on the write port these conditions must be satisfied:

- No single write bursts.

- Write burst must be greater than two data beats.
- Write burst operation must be an INCR or WRAP burst type.

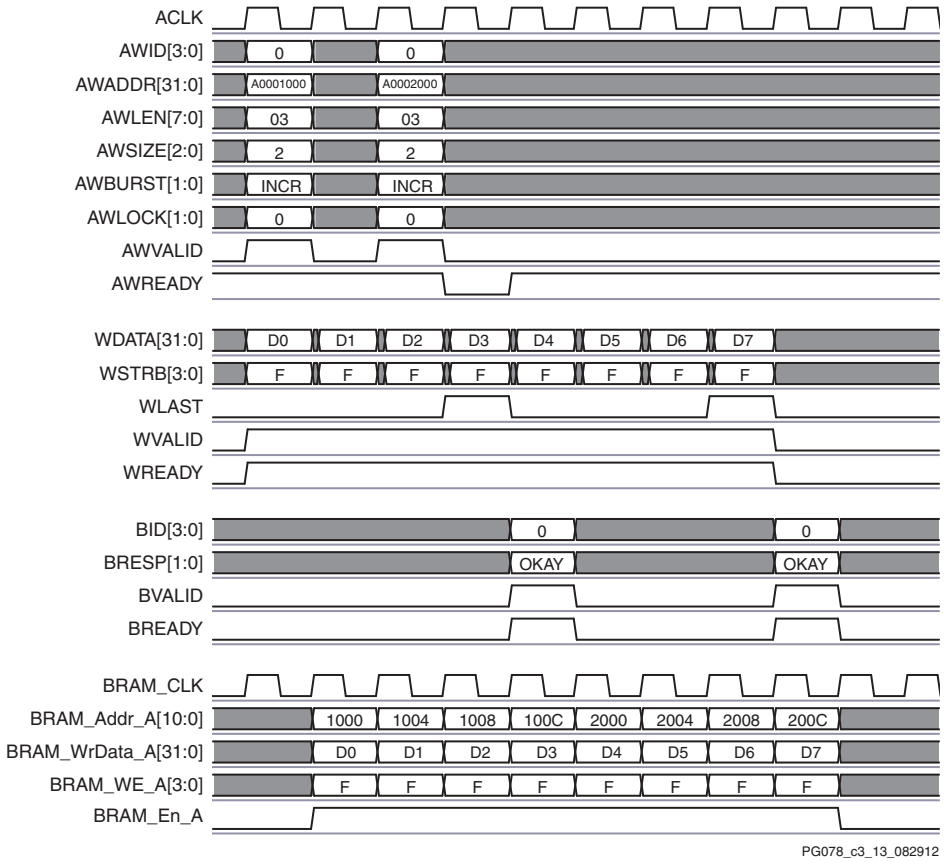

Figure 3-13: **AXI Back to Back Write Burst Timing**

Delayed Write Address

The AXI BRAM Controller throttles the write data after one data beat prior to a provided valid write address. The AXI BRAM Controller only accepts early `WVALID` and `WDATA` when configured in a dual port mode when ECC functionality is disabled. The scenario timing is shown in Figure 3-14.

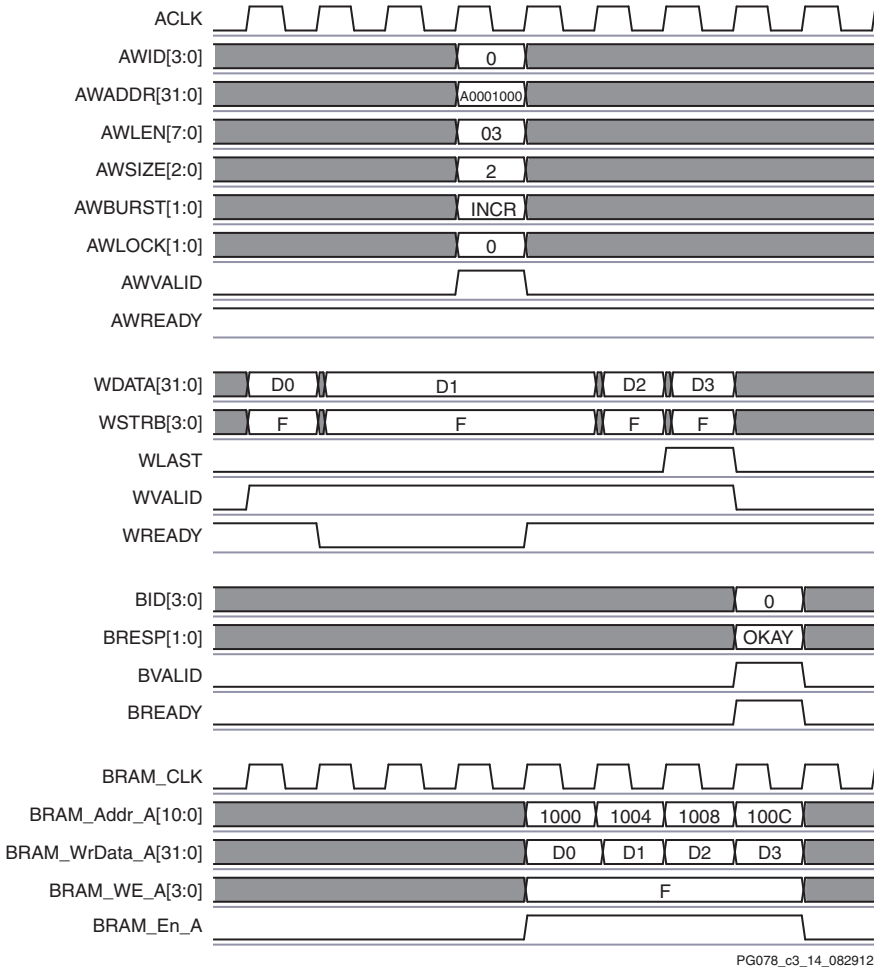

Figure 3-14: **AXI Delayed Write Address**

Read Bursting

Figure 3-15 illustrates the example timing for an AXI read burst with block RAM handled by the AXI BRAM Controller. The memory read burst starts at address 0x1000h of the block RAM, provided C_S_AXI_BASEADDR = 0xA000 0000 and C_S_AXI_HIGHADDR allows more than 4k of addressable memory. The AXI Read Address Channel interface keeps the ARREADY signal asserted until the read address pipeline is full in the AXI BRAM Controller. On the AXI Read Data Channel, the AXI BRAM Controller supports the AXI master/Interconnect to respond to the RVALID assertion with a same clock cycle assertion of RREADY. If the requesting AXI master/Interconnect throttles upon accepting the read burst data (by negating RREADY), the AXI BRAM Controller can manage this and holds the data pipeline until RREADY is asserted.

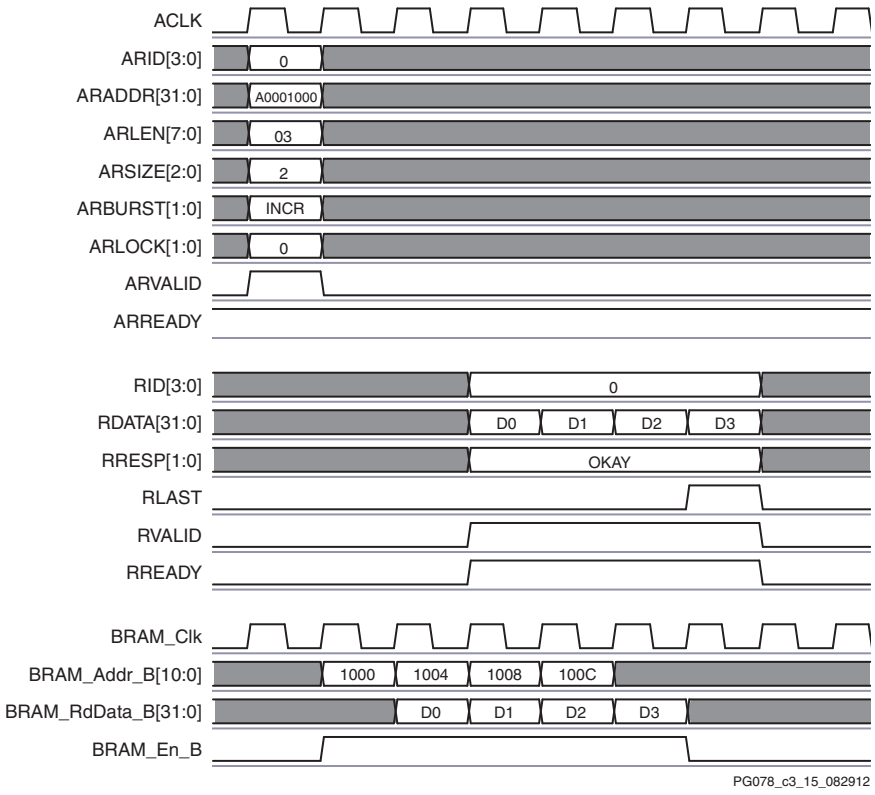

Figure 3-15: **AXI Read Burst Diagram**

Read Throttling

The AXI BRAM Controller must support read throttling. During a read operation, the AXI BRAM issues read commands to the block RAM, but can only read ahead two addresses (the amount of BRAM read data beats supported in the AXI BRAM Controller read data skid buffer pipeline). The requesting AXI master is not required to capture all the data immediately, but might throttle and only assert the RREADY signal when data can be accepted. The AXI BRAM Controller must halt the read operation and hold existing read data when the requesting master negates RREADY. Figure 3-16 illustrates this behavior and corresponding BRAM port operation. The two stage read data pipeline ensures that all outputs to block RAM and outputs to the AXI read data channel are registered.

The behavior shown in Figure 3-16 reflects the condition when the master waits to assert RREADY until RVALID is asserted. The AXI BRAM Controller can accept the master assertion of RREADY prior to the assertion of RVALID. Both signals must be asserted to advance the read data skid buffer pipeline in the AXI BRAM Controller.

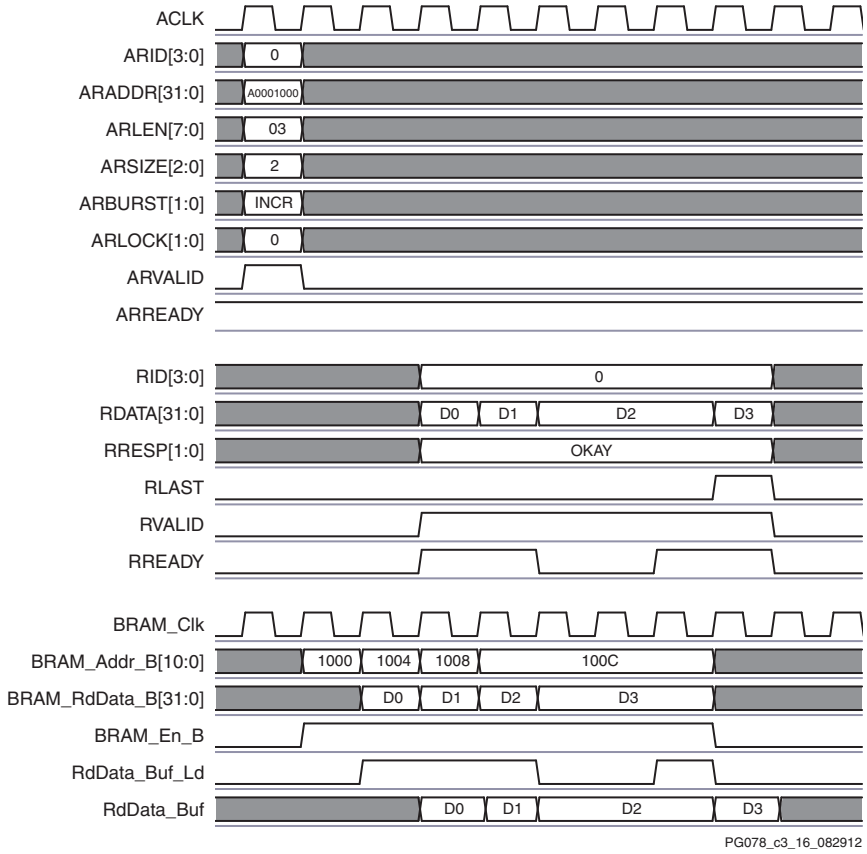

Figure 3-16: **AXI Read Burst Throttling**

Read Address Pipeline

Figure 3-17 and Figure 3-18 illustrate examples of the timing for pipelined read burst operations. The AXI BRAM Controller can handle pipelined read addresses as a continuous burst to block RAM. The master of the pipelined read operation can accept data in the clock cycle following the assertion of `RLAST` (from the prior read operation) under these conditions:

- The read operation is not a single data beat transfer
- The read burst is greater than two data beats
- The AXI burst operation size is equal to the data port size of the AXI Interconnect
- The requesting burst type is of type INCR or WRAP
- No throttling is detected (on the AXI read channel) for the current read burst after the second to last BRAM address is registered out to memory

A continuous read burst to block RAM is supported in both a dual port configuration (using the second port to block RAM), or in a single port BRAM configuration (pending no active pending write transfers).

If any of these previously cited conditions exist on the pipelined read operations, the master must wait until RVALID is reasserted to begin reading data for the subsequent burst. The expected delay is two AXI clock cycles until `RVALID` is asserted after the prior `RLAST` (when any of these above conditions exist). Figure 3-18 illustrates the timing for this scenario.

Figure 3-17 illustrates the ability of the AXI BRAM Controller to accept pipelined read request addresses and maintain 100% bus utilization to the block RAM. The data burst must be greater than two data beats to reach a maximum 100% data throughput from the block RAM with no idle clock cycles on the AXI read data channel. The requesting burst type must be INCR or WRAP and the requesting read burst size must be equal to the size of the AXI Interconnect read data port (no "narrow" burst type transactions) to achieve 100% bus utilization on pipelined read bursts.

Utilization of the read data skid buffer illustrates the master capability to throttle on accepting read data. The resulting BRAM transaction timing is shown in Figure 3-18.

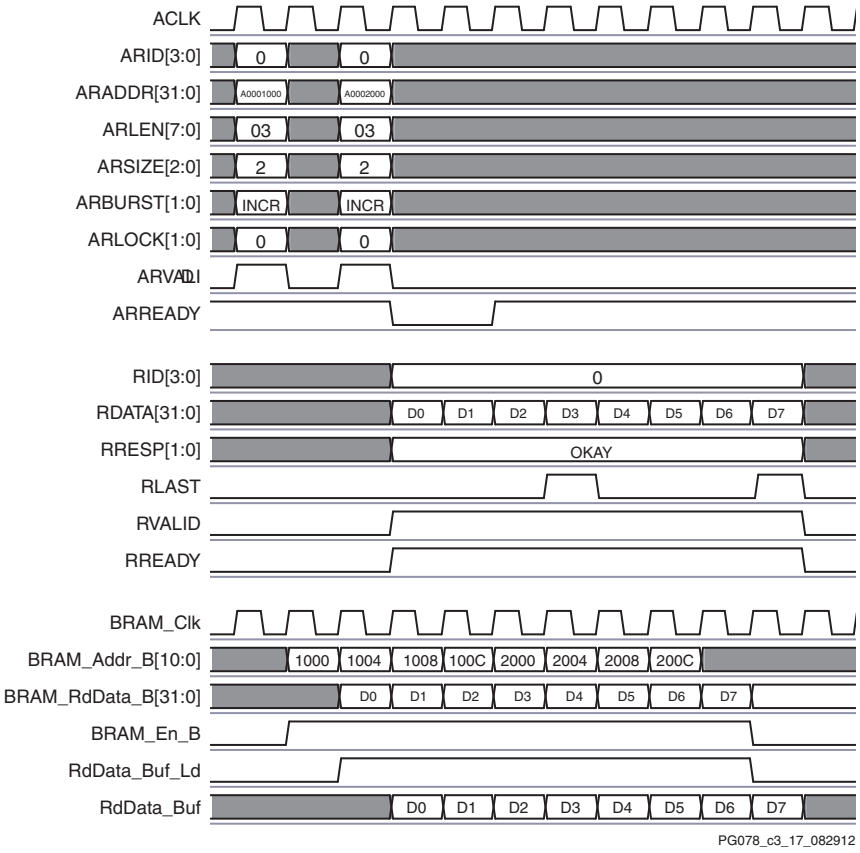

Figure 3-17: **AXI 100% Bus Utilization on Pipelined Read Bursts**

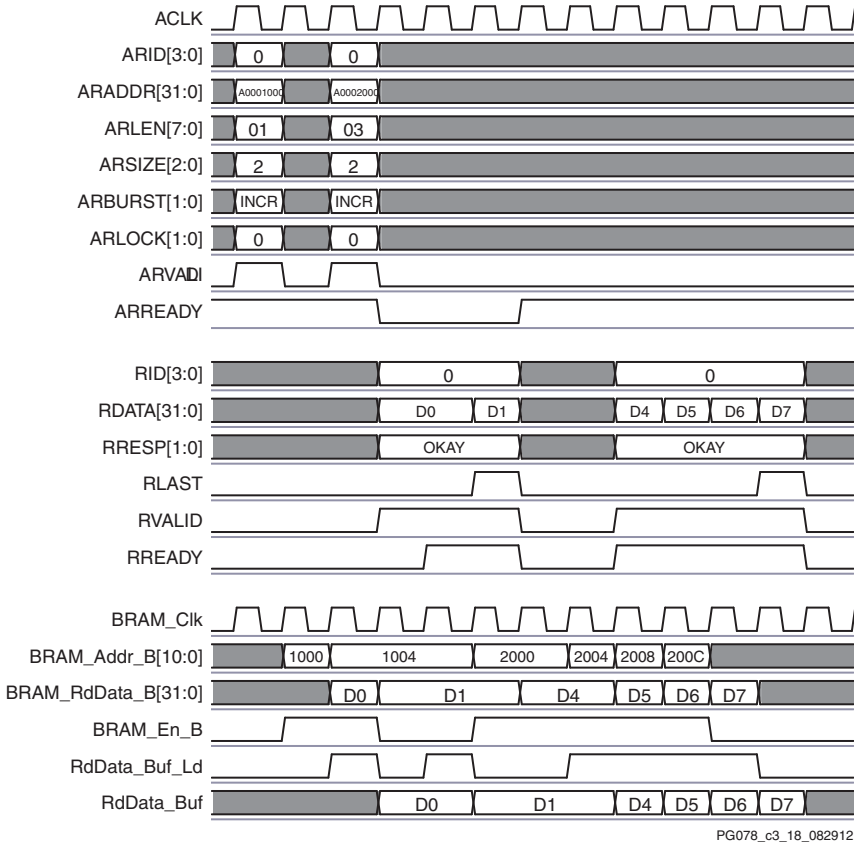

Figure 3-18: **AXI Read Pipeline Throttling Timing**

Cacheline Reads

Figure 3-19 illustrates the timing on AXI WRAP or cacheline burst transactions. The address generated to the block RAM starts at the target word and wraps around once the address boundary is reached.

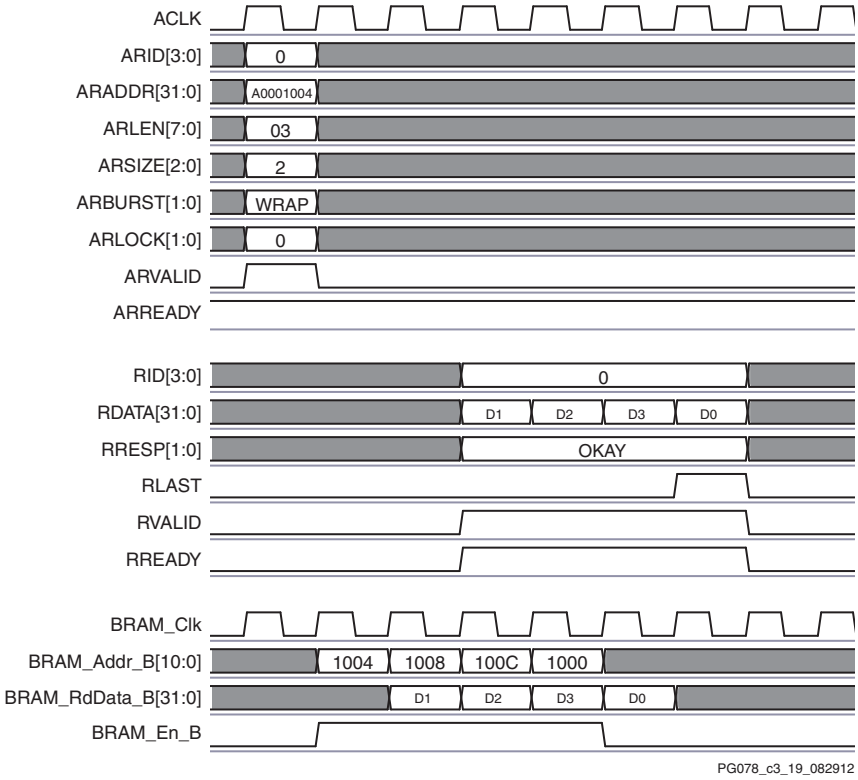

Figure 3-19: **AXI Cacheline Read**

Dual Port BRAM Capability

Isolating the write and read ports to block RAM with each write and read channel interface on the AXI bus, both BRAM ports can be utilized simultaneously. Figure 3-20 illustrates this condition to provide no arbitration on the write or read channels dependency on the other.

Single Port BRAM Capability

The AXI BRAM Controller can be configured (on an AXI4 interface) to use only a single port to block RAM. In this configuration, the AXI BRAM arbitrates on the AR and AW channels and creates a single two-stage pipeline. The arbitration scheme uses a least recently used algorithm on ties of assertions between the `S_AXI_ARVALID` and `S_AXI_AWVALID` signals. For the illustration shown in Figure 3-21, the AW channel won arbitration and accesses the block RAM first.

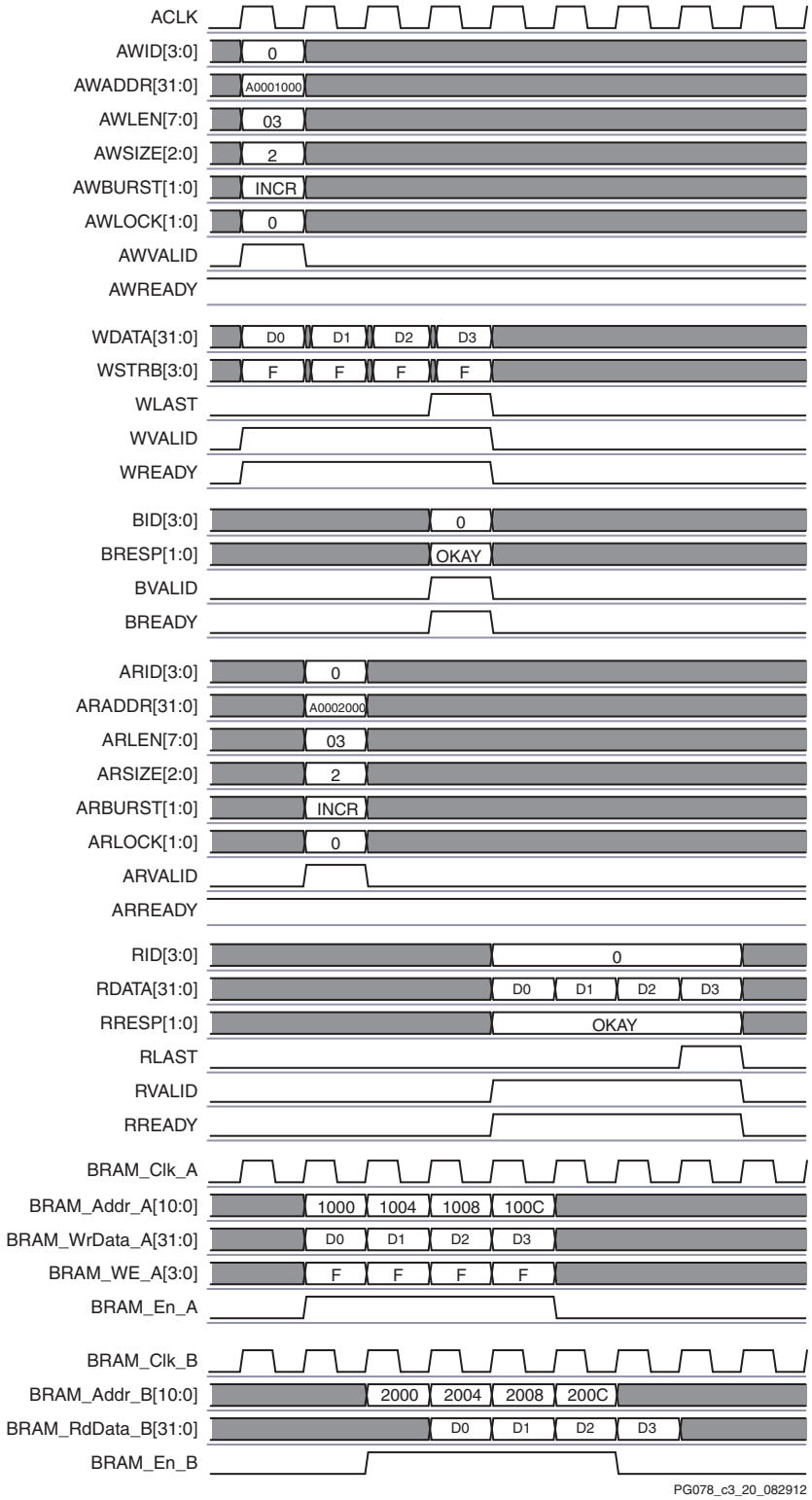

Figure 3-20: **AXI Dual Port BRAM Capability**

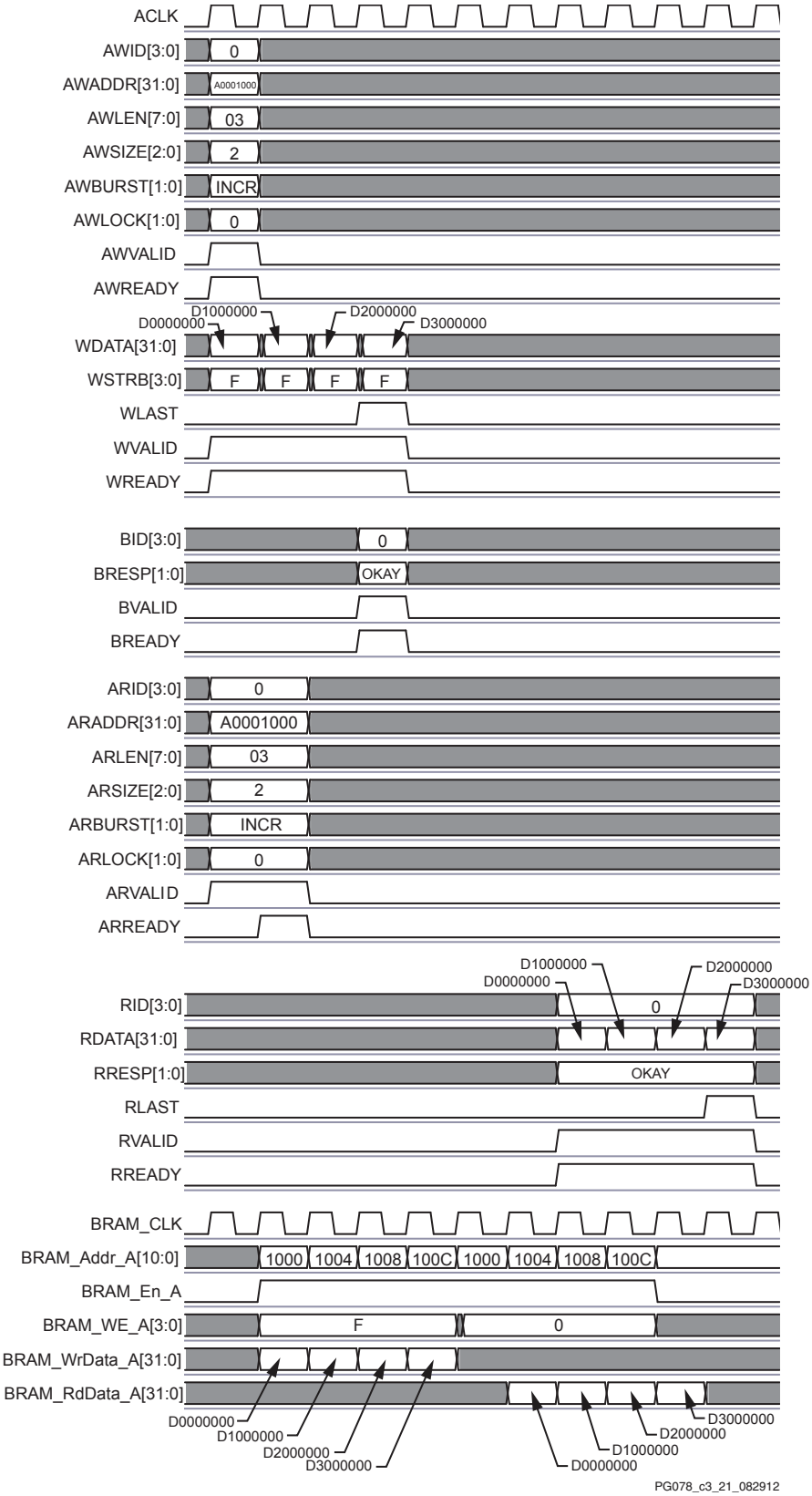

Figure 3-21: **AXI4 Single Port BRAM Capability**

Addressing Configurations

Table 3-1 illustrates the BRAM configuration (and corresponding address assignment) for targeting UltraScale, 7 series or Zynq-7000 SoC All Programmable device designs using the 36 Kb BRAM module.

Table 3-1: **BRAM Configuration for UltraScale, 7 Series, or Zynq-7000 Devices**

Supported Memory Sizes / BRAM Memory Configuration	Number of BRAM Primitives (36k/each)	Size of BRAM_Addr (each port)	Typical BRAM_Addr Bit Usage with BRAM Configuration
32-bit Data BRAM Data Width			
4k / (1024 x 32)	1	10	BRAM_Addr [11:2]
8k / (2048 x 32)	2	11	BRAM_Addr [12:2]
16k / (4096 x 32)	4	12	BRAM_Addr [13:2]
32k / (8192 x 32)	8	13	BRAM_Addr [14:2]
64k / (16384 x 32)	16	14	BRAM_Addr [15:2]
128K / (32,768x32)	32	15	BRAM_Addr [16:2]
64-bit Data BRAM Data Width			
8k / (1024 x 64)	2	10	BRAM_Addr [12:3]
16k / (2048 x 64)	4	11	BRAM_Addr [13:3]
32k / (4096 x 64)	8	12	BRAM_Addr [14:3]
64k / (8192 x 64)	16	13	BRAM_Addr [15:3]
128k / (16384 x 64)	32	14	BRAM_Addr [16:3]
128-bit Data BRAM Data Width			
16k / (1024 x 128)	4	10	BRAM_Addr [13:4]
32k / (2048 x 128)	8	11	BRAM_Addr [14:4]
64k / (4096 x 128)	16	12	BRAM_Addr [15:4]
128k / (8192 x 128)	32	13	BRAM_Addr [16:4]

Data Organization

The BRAM interface is designed to connect to the BRAM memory in a little-endian data structure, in matching the AXI data structure. The byte ordering and numbering for a 32-bit BRAM is shown in Table 3-2 and Table 3-3 for a 64-bit BRAM.

Table 3-2: **Little Endian Byte Ordering (32-bit BRAM)**

Byte Address	n+3	n+2	n+1	n
Byte Label	3	2	1	0
Byte Significance	MSByte			LSByte

Table 3-2: **Little Endian Byte Ordering (32-bit BRAM) *(Cont'd)***

Bit Label	31		0
Bit Significance	MSBit		LSBit

Table 3-3: **Little Endian Byte Ordering (64-bit BRAM)**

Byte Address	n+7	n+6	...	n+1	n
Byte Label	7	6	...	1	0
Byte Significance	MSByte		...		LSByte
Bit Label	63				0
Bit Significance	MSBit				LSBit

ECC

To mitigate the effect of BRAM Single Event Upsets, SEUs, the AXI BRAM Controller can be configured to use Error Correction Codes, ECC. When writing to the block RAM, ECC bits are generated and stored together with the written data. When reading from the block RAM, the ECC bits are used to correct any single bit errors and detect any double bit errors in the data read. Errors are either signaled on the AXI connection, AXI_RRESP (a SLVERR value) to MicroBlaze™ processor (or the requesting AXI master device), or through an interrupt signal.

ECC is available in both AXI4-Lite and AXI4 interface configurations of the AXI BRAM Controller core, and for both single and dual port connection types to the BRAM block (the setting of C_SINGLE_PORT_BRAM can be 0 or 1). ECC is supported on BRAM block data widths of 32, 64, or 128 bits only. The only combination not allowed for ECC support is when C_S_AXI_DATA_WIDTH = 256 or larger data widths.

The available ECC control and status registers described in (ECC Register Details, page 24) are available when C_ECC = 1.

ECC is inserted on the write datapath in the AXI BRAM Controller for the data presented to the BRAM block at each memory location. In a full AXI interface connection to support all data transaction types (including narrow and unaligned data transfers, the ECC logic incorporates a read-modify-write sequence. The design ensures that each write to block RAM stores the full data width, with the supporting ECC data bits. The AXI BRAM Controller supports 32-bits of data, with 7 bits of ECC data generated. In a 64-bit configuration, 8 bits of ECC data are created to store with the corresponding data. For 128-bit BRAM data width with ECC requires 9 bits of ECC to be stored with the data.

Each set of ECC bits are based (and calculated) on the full word (for 32-bit BRAM configurations) or the full double word (in 64-bit BRAM configurations). For each AXI transaction, a read-modify-write sequence is performed to read the full word (or BRAM data width) of data, merge in the new write data (indicated by the AXI_WSTRB signal) on the AXI bus, and the write the new data (and ECC bits) to block RAM. This sequence is shown in

Figure 3-4, page 38 (on the AXI4-Lite interface) and Figure 3-11, page 49 (for the AXI4 interface).

Table 3-4 summarizes the additional ECC data bits to the BRAM interface.

Table 3-4: **ECC Data Sizes**

C_S_AXI_DATA_WIDTH	Number of ECC data bits	Number of ECC WEs
32	7	1
64	8	1
128	9	2

A hamming code algorithm is used to generate the ECC bits for 32-bit and 64-bit BRAM data widths. An HSIAO algorithm is used to generate the ECC bits for 32-bit, 64-bit and 128-bit BRAM data widths.

The design parameter, C_ECC_ONOFF_RESET_VALUE (default on), determines the reset value for the enable/disable setting of ECC. ECC checking is on by default and can turned off during reset.

Hsiao ECC

The Hsiao algorithm, which is utilized in the Xilinx MIG HDL and MPMC tools is incorporated in the AXI BRAM Controller. The Hsiao algorithm provides an ECC encoding/decoding for optimized resource utilization and performance. The Hsiao algorithm is designed such that an equal number of inputs are used to generate each check bit in the ECC code. The number of inputs to generate each check bit depends on the overall data width of the block RAM + required number of ECC bits for that data width.

The AXI BRAM HDL generates the h-matrix as shown in Figure 3-22 for 32-bit BRAM data width configurations. The same methodology is applied to 64-bit and 128-bit data width configurations with ECC enabled. The h-matrix follows the rules of the Hsiao ECC algorithm. See the IBM documentation by M.Y. Hsiao in References in Appendix D.

The dimensions of the h-matrix is determined by the number of ECC bits and the overall code width (which equals the BRAM data width + ECC code width). For 32-bit systems, the h-matrix is 7 x 39. And filled by combinations of (7 1) for the ECC bits, plus combinations of (7 3). Not all (7 3) combinations are used in the 32 data bits of the h-matrix.

For 64-bit systems, the h-matrix is 8 x 72. The h-matrix is constructed of all the (8 1) combinations for the ECC bits, all (7 3) combinations are used, and 8 combinations of the weight-5 codes (8 5) are used.

The Hsiao generator creates the "one at a time" column entries for the ECC parity bits. The generator then starts at one end of the h-matrix (shown as the last entry in Figure 3-22), and generates the combination of weight-3 codes. For 32-bit data widths, not all weight-3

codes are used. For 64-bit h-matrix generation, when all weight-3 codes are used, it generates weight-5 codes.

The previously described ECC encoding is not unique; there are many other valid SEC-DEC codes that can be utilized. Special consideration must be applied to systems in which port B of the block RAM is used by a separate system or a different AXI BRAM Controller. The ECC algorithms for encoding and decoding data must be identical between both ports.

For ECC bit generation, the column of the h-matrix is merged with the data pattern written to memory. The h-matrix bits pertaining to each check bit are XOR together to generate each check bit.

Upon ECC decoding, the entire h-matrix (data bits + ECC bits) are used to generate the syndrome value. For Hsiao ECC, when the syndrome value is equal to all zeros, no errors exist in the data. When the syndrome value has at least one bit equal to '1', and the number of 1's in the syndrome is an odd number, then a single bit error is detected. The syndrome value that matches the h-matrix indicates the bit position in error. When the syndrome value has at least one bit equal to '1', and the number of 1's in the syndrome is an even number, then a double bit error is detected.

The parity check matrix for 32-bit BRAM data (39, 32) Hsiao code is shown in Figure 3-22.

H-Matrix Bit	H-Matrix Value (Hex) (6:0)	H-Matrix Value (Binary) (6:0)	ECC Bit Positions Set to '1'	Notes
38	01	0000001	1	
37	02	0000010	2	
36	04	0000100	3	
35	08	0001000	4	
34	10	0010000	5	
33	20	0100000	6	
32	40	1000000	7	When the h-matrix reaches the data width, only one bit in the the ECC word is asserted.
31	0E	0001110	234	
30	13	0010011	125	
29	15	0010101	135	
28	16	0010110	235	
27	19	0011001	145	
26	1A	0011010	245	
25	1C	0011100	345	
24	23	0100011	126	
23	25	0100101	136	
22	26	0100110	236	
21	29	0101001	146	
20	2A	0101010	246	
19	2C	0101100	346	
18	31	0110001	156	
17	32	0110010	256	
16	34	0110100	356	
15	38	0111000	456	
14	43	1000011	127	
13	45	1000101	137	
12	46	1000110	237	
11	40	1001001	147	
10	4A	1001010	247	
9	4C	1001100	347	
8	51	1010001	157	
7	52	1010010	257	
6	54	1010100	357	
5	58	1011000	457	
4	61	1100001	167	
3	62	1100010	267	And so on...
2	64	1100100	367	Again, finds next combination based on prior seed.
1	68	1101000	467	Generates “1101000” based on initial seed of prior combination, “111000”.
0	70	1110000	567	Starting point in H-matrix calculation. Three unique bits of the ECC word are asserted.

PG078_c3_01_082812

Figure 3-22: **32-bit Hsiao Encoding**

The h-matrix generated for 64-bit ECC encoding in shown in Figure 3-23.

H-Matrix Bit	H-Matrix Value (Hex) (7:0)	H-Matrix Value (Binary) (7:0)	Notes	H-Matrix Bit	H-Matrix Value (Hex) (7:0)	H-Matrix Value (Binary) (7:0)	Notes
71	01	00000001		39	31	00110001	
70	02	00000010		...			
69	04	00000100		31	4A	01001000	
68	08	00001000		30	4C	01001100	
67	10	00010000		29	51	01010001	
66	20	00100000		28	52	01010010	
65	40	01000000	When the H-matrix reaches the data width maximum, only one bit of the ECC is asserted.	27	54	01010100	
				26	58	01011000	
64	80	10000000		25	61	01100001	
63	E6	11100110		24	62	01100010	
62	E9	11101001		23	64	01100100	
61	Ea	11101010		22	68	01101000	
60	Ec	11101100		21	70	01110000	
59	F1	11110001		20	83	10000011	
58	F2	11110010		19	85	10000101	
57	F4	11110100	Starts with weight-5 codes. Some implementations of Hsiao dispurse the weight-5 codes with the weight-3 codes, but they are assigned to the upper order data bits.	18	86	10000110	
56	F8	11111000		17	89	10001001	
55	07	00000111		16	8A	10001010	
54	0B	00001011		15	8C	10001100	
53	0d	00001101	64-bit Hsiao uses all weight-3 code combinations.	14	91	10010001	
52	0E	00001110		13	92	10010010	
51	13	00010011		12	94	10010100	
50	15	00010101		11	98	10011000	
49	16	00010110		10	A1	10100001	
48	19	00011001		9	A2	10100010	
47	1A	00011010		8	A4	10100100	
46	1C	00011100		7	A8	10101000	
45	23	00100011		6	B0	10110000	
44	25	00100101		5	C1	11000001	
43	26	00100110		4	C3	11000010	
42	29	00101001		3	C4	11001000	And so on...
41	2A	00101010		2	C8	11001000	Again, finds next combination based on prior seed.
40	2C	00101100		1	D0	11010000	Generates “1101000” based on initial seed of prior combination, “111000”.
				0	E0	11100000	Starting point in H-matrix calculation.

PG078_c3_02_082812

Figure 3-23: **64-bit Hsiao Encoding**

The Hsiao ECC encoding (137, 128) for 128-bit configurations is shown in Figure 3-24. Not all encodings are shown, but the concept is identical to the 32-bit and 64-bit implementations. The 128-bit ECC algorithm is shown in Figure 3-24.

H-Matrix Bit	H-Matrix Value (Hex) (8:0)	H-Matrix Value Binary (8:0)	Notes
136	001	000000001	
135	002	000000010	
134	004	000000100	
133	008	000001000	
132	010	000010000	
131	020	000100000	
130	040	001000000	
129	080	010000000	
128	100	100000000	When the H-matrix reaches the data width maximum, the ECC bits have one bit asserted.
127	165	101100101	
126	166	101100110	
125	169	101101001	
124	16C	101101100	
123	171	101110001	
122	172	101110010	
121	174	101110100	
120	178	101111000	Weight-5 codes
119	187	110000111	
118 117	18B 18D	110001011 110001101	
116	18E	110001110	
...			
85	1E8	111101000	
84	1F0	111110000	Start of weight-5 codes
83	007	000000111	Last combination of a weight-3 code
...			
7	160	101100000	
6	181	110000001	
5	182	110000010	
4	184	110000100	
3	188	110001000	
2	190	110010000	
1	1A0	110100000	
0	1C0	111000000	Start with weight-3 code (MSB all 1s)

PG078_c3_03_082812

Figure 3-24: **128-bit Hsiao Encoding**

Hamming Code

The ECC supports the BRAM data widths of 32-bit and 64-bits. The ECC used is a (32, 7) Hamming code with the coding defined by Table 3-5.

Table 3-5: **ECC 32-bit Encoding**

Participating Data Bits	ECC0	ECC1	ECC2	ECC3	ECC4	ECC5	ECC6
0	•	•					•
1	•		•				•
2		•	•				•

Table 3-5: **ECC 32-bit Encoding**

Participating Data Bits	ECC0	ECC1	ECC2	ECC3	ECC4	ECC5	ECC6
3	•	•	•				
4	•			•			•
5		•		•			•
6	•	•		•			
7			•	•			•
8	•		•	•			
9		•	•	•			
10	•	•	•	•			•
11	•				•		•
12		•			•		•
13	•	•			•		
14			•		•		•
15	•		•		•		
16		•	•		•		
17	•	•	•		•		•
18				•	•		•
19	•			•	•		
20		•		•	•		
21	•	•		•	•		•
22			•	•	•		
23	•		•	•	•		•
24		•	•	•	•		•
25	•	•	•	•	•		
26	•					•	•
27		•				•	•
28	•	•				•	
29			•			•	•
30	•		•			•	
31		•	•			•	

The ECC used in the AXI BRAM controller when C_S_AXI_DATA_WIDTH is set to 64 is based on (64, 8) Hamming code derived from the (32, 7) 32-bit hamming code methodology. The generation of the 8-bit ECC check bits for a 64-bit BRAM data width is shown in Table 3-6.

Table 3-6: **ECC 64-bit Encoding**

Participating Data Bits	ECC0	ECC1	ECC2	ECC3	ECC4	ECC5	ECC6	ECC7 (Overall Parity = Inverted)
0-31	Same as shown in Table 3-5.						Set to '0'	Holds ECC6 bit settings from Table 3-5.
32	•	•	•			•		•
33				•		•		•
34	•			•		•		
35		•		•		•		
36	•	•		•		•		•
37			•	•		•		
38	•		•	•		•		•
39		•	•	•		•		•
40	•	•	•	•		•		
41					•	•		•
42	•				•	•		
43		•			•	•		
44	•	•			•	•		•
45			•		•	•		
46	•		•		•	•		•
47		•	•		•	•		•
48	•	•	•		•	•		
49				•	•	•		
50	•			•	•	•		•
51		•		•	•	•		•
52	•	•		•	•	•		
53			•	•	•	•		•
54	•		•	•	•	•		
55		•	•	•	•	•		
56	•	•	•	•	•	•		•
57	•						•	•
58		•					•	•
59	•	•					•	
60			•				•	•
61	•		•				•	

Table 3-6: **ECC 64-bit Encoding**

Participating Data Bits	ECC0	ECC1	ECC2	ECC3	ECC4	ECC5	ECC6	ECC7 (Overall Parity = Inverted)
62		•	•				•	
63	•	•	•				•	•

Clocking

The AXI BRAM Controller is fully synchronous with all clocked elements clocked with S_AXI_ACLK. The BRAM_Clk_A is the output clock used for clocking the BRAM Interface, which is connected to S_AXI_ACLK with same frequency and phase alignment.

Resets

The S_AXI_ARESETN is the master reset input signal for the AXI BRAM Controller used in both S_AXI and S_AXI_CTRL. The BRAM_Rst_A output signal is used to drive the BRAM Interface.

Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 3]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 8]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 9]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 10]

Customizing and Generating the Core

This section includes information about using Xilinx tools to customize and generate the core in the Vivado Design Suite environment.

Vivado Integrated Design Environment (IDE)

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu .

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 8] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 9].

Note: Figures in this chapter are illustrations of the Vivado IDE. This layout might vary from the current version.

The AXI BRAM Controller parameters are divided in three categories within the IDE: General Protocol options, BRAM options and ECC Options.

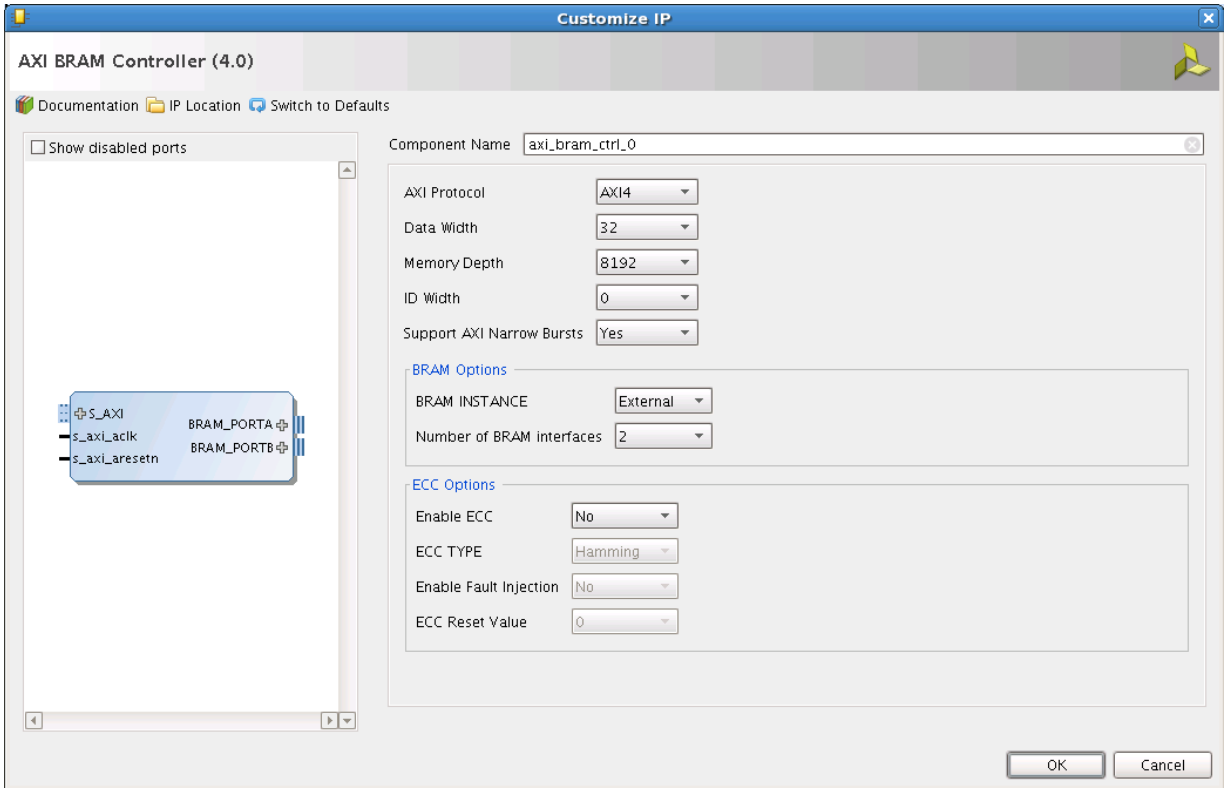

Figure 4-1: **Customize IP - AXI BRAM Controller**

General Protocol Options

- **AXI Protocol**: AXI protocol connection type: AXI4, AXI3 or AXI4-Lite.
- **Data Width**: Data width of AXI slave (BRAM data width): 32, 64, 128, 256, 512 or 1024 bits. The maximum data width supported is 1024 bits.
- **Memory Depth**: Memory depth of AXI Slave: 1k, 2k, 4k, 8k, 16k, 32k, 64k, 128k, or 256k words. The maximum memory depth supported is 128K words. See Table 1-1 for more details.

  Note: When using IP integrator, the value of this parameter is automatically propagated during the design validation from the master.

  In IP Integrator, AXI BRAM Controller supports the following Memory Width and Depth values.

Table 4-1: **Memory Width and Depth values in IP Integrator**

Memory Width	Memory Depth
less than or equal to 128	1024,2048,4096,8192,16384,32768,65536,131072,262144,524288,1048576,2097152,4194304,8388608,16777216,33554432,67108864,134217728,268435456,536870912,1073741824
greater than 128 and less than or equal to 256	1024,2048,4096,8192,16384,32768,65536,131072,262144,524288,1048576,2097152,4194304,8388608,16777216,33554432,67108864,134217728,268435456,536870912
greater than 256 and less than or equal to 512	1024,2048,4096,8192,16384,32768,65536,131072,262144,524288,1048576,2097152,4194304,8388608,16777216,33554432,67108864,134217728,268435456
greater than 512 and less than or equal to 1024	1024,2048,4096,8192,16384,32768,65536,131072,262144,524288,1048576,2097152,4194304,8388608,16777216,33554432,67108864,134217728
greater than 1024 and less than or equal to 2048	1024,2048,4096,8192,16384,32768,65536,131072,262144,524288,1048576,2097152,4194304,8388608,16777216,33554432,67108864

AXI BRAM Controller supports the following memory width and memory depth values when connected to Block Memory Generator IP.

Table 4-2: **Memory Width and Depth values when connected to Block Memory Generator IP**

Memory Width	Memory Depth
less than or equal to 128	1024,2048,4096,8192,16384,32768,65536,131072,262144
greater than 128 and less than or equal to 256	1024,2048,4096,8192,16384,32768,65536,131072
greater than 256 and less than or equal to 512	1024,2048,4096,8192,16384,32768,65536
greater than 512 and less than or equal to 1024	1024,2048,4096,8192,16384,32768
greater than 1024 and less than or equal to 2048	1024,2048,4096,8192,16384

- **ID Width**: Width of ID vectors in AXI system (includes all master and slave devices). The range of ID width supported is 0 to 32.

  Note: When using IP integrator, the value of this parameter is automatically propagated during the design validation from the master.

- **Support AXI Narrow Bursts**: Support of AXI4 narrow write or read operations.

  Note: When using IP Integrator, you can set the value of this parameter or set it to **Auto**, in which the value is propagated from the master.

- **BRAM Instance**: Specifies whether the BRAMs are available internally (with the AXI BRAM CTRL IP) or externally.

  Note: When using IP integrator, the value of the BRAM Instance parameter is set to **External**. The Number of BRAM Interfaces can be set to either single port (1) or dual port (2) BRAM Interface.

- **ECC Options**:
  - Enable ECC: Enables or disables the ECC checking.
  - ECC Type: Select the ECC algorithm to be used: **Hamming** (default) or **HSIAO**. This option is available only when **Enable ECC** is **Yes**.
  - Enable Fault Injection: Specifies whether the fault is injected. When this option is **Yes**, the errors can be injected in the data as well as in the generated ECC written to the block RAM. This option is available only when **Enable ECC** is **Yes**.
  - ECC Reset Value: Determines the reset value for the enable/disable setting of ECC. ECC checking can be turned on/off during reset using this option. This option is available only when **Enable ECC** is **Yes**.

Output Generation

For details, see "Generating IP Output Products" in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 8].

Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

Required Constraints

There are no required constraints for this core.

Device, Package, and Speed Grade Selections

The target technology is an FPGA listed in the Supported Device Family field in the Vivado IP Catalog.

Clock Frequencies

See Performance in Chapter 2 for information about clock frequencies.

Clock Management

The AXI BRAM Controller is fully synchronous with all clocked elements clocked by the `s_axi_aclk` input.

For MicroBlaze™ processor operation, the `s_axi_aclk` must be the same as the MicroBlaze clock.

Clock Placement

There are no clock placement requirements for this core.

Banking

There are no banking requirements for this core.

Transceiver Placement

There are no transceiver requirements for this core.

I/O Standard and Placement

There are no I/O requirements for this core.

Simulation

This section contains information about simulating IP in the Vivado Design Suite.For details, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 10].

Synthesis and Implementation

This section contains information about synthesis and implementation in the Vivado Design Suite.

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 8].

Test Bench

This chapter contains information about the test bench provided in the Vivado® Design Suite.Figure 5-1 shows a block diagram of the demonstration test bench.

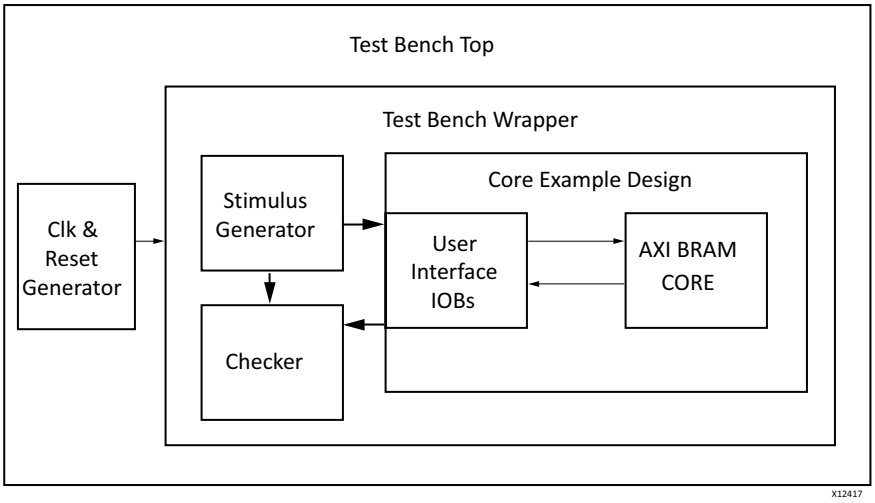

Figure 5-1: **Test Bench Block Diagram**

The demonstration test bench is a straightforward VHDL file to exercise the example design and the core itself. The test bench consists of the following:

- Clock generators
- Stimulus generator module which includes the address and data generator module
- Data checker
- Protocol checks for the AXI4 protocol

The demonstration test bench in a core with an AXI4 interface performs the following tasks:

- Input clock signals are generated.
- A reset is applied to the example design.
- Stimulus is generated only for incremented burst, and narrow transactions are not exercised.
- Address is generated in a linear incremented format.
- Pseudo-random data is generated for WDATA.

- Length (AWLEN/ARLEN) is determined randomly.
- During read, RDATA is compared with another pseudo-random generator with the same seed as WDATA random data generator.
- A basic AXI4 protocol checker checks the protocol violations.
- Core is exercised with 256 AXI4 write and read transactions.

Messages and Warnings

When the functional or timing simulation has completed successfully, the test bench displays the following message, and it is safe to ignore this message.

```
Failure: Test Completed Successfully
```

Example Design

This chapter contains information about the example design provided in the Vivado® Design Suite.The AXI BRAM Controller example design includes the following:

- HDL wrapper that instantiates the AXI BRAM Controller netlist
- AXI BRAM Controller constraint file

The AXI BRAM Controller example design has been tested with the Vivado Design Suite.

Migrating and Upgrading

This appendix contains information about migrating a design from ISE® to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado Design Suite, important details (where applicable) about any port changes and other impact to user logic are included.

Migrating to the Vivado Design Suite

For information about migrating to the Vivado Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 11].

Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado Design Suite.

Parameter Changes

The user parameter "ADDR_WIDTH" has been removed.

Port Changes

The ID ports (`s_axi_arid`, `s_axi_awid`, `s_axi_bid`, and `s_axi_rid`) are not generated when the ID width is set to 0.

XILINX
ALL PROGRAMMABLE™

Appendix B

Verification, Compliance, and Interoperability

Simulation

The AXI BRAM Controller core delivered is rigorously verified using advanced verification techniques, including a constrained random configuration generator.

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

Finding Help on Xilinx.com

To help in the design and debug process when using the AXI BRAM Controller, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the AXI BRAM Controller. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Design Tools tab on the Downloads page. For more information about this tool and the features available, open the online help after installation.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core are listed below, and can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record

AR: 54418

Technical Support

Xilinx provides technical support in the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

Xilinx provides premier technical support for customers encountering issues that require additional assistance.

To contact Xilinx Technical Support, navigate to the Xilinx Support web page.

Note: Access to WebCase is not available in all cases. Login to the WebCase tool to see your specific support options.

Debug Tools

There are many tools available to address AXI BRAM Controller design issues. It is important to know which tools are useful for debugging various situations.

Vivado Design Suite Debug Feature

Vivado Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. This feature allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature represents the functionality in the Vivado IDE that is used for logic debugging and validation of a design running in Xilinx FPGA devices in hardware.

The Vivado logic analyzer is used to interact with the logic debug LogiCORE IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 12].

Reference Boards

Various Xilinx development boards support theAXI BRAM Controller. These boards can be used to prototype designs and establish that the core can communicate with the system.

- 7 series FPGA evaluation boards
  - KC705
  - KC724

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The ChipScope debugging tool is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the ChipScope debugging tool for debugging the specific problems.

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If your outputs go to 0, check your licensing.

Interface Debug

AXI4-Lite Interfaces

Read from a register that does not have all 0s as a default to verify that the interface is functional. See Figure 3-3 for a read timing diagram. Output `s_axi_arready` asserts when the read address is valid, and output `s_axi_rvalid` asserts when the read data/response is valid. If the interface is unresponsive, ensure that the following conditions are met:

- The `S_AXI_ACLK and ACLK` inputs are connected and toggling.
- The interface is not being held in reset, and `S_AXI_ARESET` is an active-Low reset.

Appendix D

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

References

These documents provide supplemental material useful with this product guide:

1. *AMBA AXI4-Stream Protocol Specification*
2. *LogiCORE IP AXI Interconnect Data Sheet* (DS768)
3. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
4. *7 Series FPGA Overview* (DS180)
5. *7 Series FPGAs Memory Interface Solutions User Guide* (UG586)
6. IBM Technical Journal. M. Y. Hsiao. *A Class of Optimal Minimum Odd-weight-column SEC=DED Codes*. July 1970.
7. *Vivado Design Suite Migration Methodology Guide* (UG911)
8. *Vivado Design Suite User Guide: Designing with IP* (UG896)
9. *Vivado Design Suite User Guide: Getting Started* (UG910)
10. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
11. *ISE to Vivado Design Suite Migration Methodology Guide* (UG911)
12. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/30/2015	4.0	Address range supported is increased to 2G in IP Integrator.
04/02/2014	4.0	Updated the behavior of the ID ports (s_axi_arid, s_axi_awid, s_axi_bid, and s_axi_rid) when the ID width is set to 0.
12/18/2013	3.0	• Added support of Hamming code for the BRAM data widths of 32 and 64-bits. • Added support for UltraScale™ devices.
10/02/2013	3.0	• Added details about the Supported Memory Size in Chapter 1. • Changed signal names to all use lowercase. • Updated the data in Table 2-1. • Added ECC options in the Vivado IDE. • Added Simulation, Synthesis and Implementation, Example Design and Test Bench chapters. • Added support for IP integrator.
03/20/2013	3.0	Updated core version. Updated core registers.
12/18/2012	2.0	Updated for Vivado Design Suite 2012.4. Added BRAM Instance option in the GUI description in Chapter 4, Customizing and Generating the Core.
10/16/2012	1.0	Initial Xilinx release as a product guide. Replaces DS777, *LogiCORE IP AXI BRAM Controller Data Sheet*. • Updated for Vivado Design Suite 2012.3. • Removed support for ISE Design Suite Embedded Edition.

Please Read: Important Legal Notices